UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Biomèdica

PLATAFORMA PER LA VISUALITZACIÓ I ANÀLISI D'IMATGES D'ANATOMIA PATOLÒGICA

Memòria i Annexos

Autor: Sandra Bernaus Tomé
Director: Raúl Benítez Iglèsias

Convocatòria: Juny 2019

Resum

Aquest projecte consisteix en el desenvolupament d'un software destinat a anatomia patològica, amb l'objectiu de proporcionar una eina de visualització, processament i anàlisi d'imatges biomèdiques a centres de recerca de malalties rares. La finalitat de la interfície gràfica d'usuaris (GUI) que s'ha dissenyat, és el suport en la recerca de patrons en els teixits malalts, per poder conèixer les malalties i ampliar el coneixement sobre aquestes.

Per a la planificació del projecte s'ha seguit la metodologia en cascada, on les etapes es comencen una darrere de l'altra. En primer lloc, s'ha fet una selecció del llenguatge i tipus de programació, continuant per una anàlisi de les tasques a realitzar, la gestió econòmica, el disseny (UML), el desenvolupament i finalment la validació. La GUI ha estat desenvolupada amb el llenguatge Python utilitzant la programació orientada a objectes i llibreries com ara PyQt5 per al disseny de la interfície, Scikit-image per al processament d'imatges i Matplotlib per a la representació i interacció entre l'usuari i les imatges.

El software permet a l'usuari seleccionar la imatge que desitgi, triar entre diferents processaments i segmentacions per aplicar a la imatge. En cas de fer una segmentació ens permet visualitzar les característiques de cada regió en una taula, on es relacionarà cada etiqueta amb les seves propietats. A més a més és possible visualitzar les propietats de cada regió en gràfiques, de manera que les propietats es poden comparar i trobar discordances entre elles.

Resumen

Este proyecto consiste en el desarrollo de un software destinado a anatomía patológica, con el objetivo de proporcionar una herramienta de visualización, procesamiento y análisis de imágenes biomédicas en centros de investigación de enfermedades raras. La finalidad de la interfaz gráfica de usuarios (GUI) que se ha diseñado, es la búsqueda de patrones en los tejidos enfermos, para poder conocer las enfermedades y ampliar el conocimiento que se tiene sobre estas.

Para la planificación del proyecto se ha seguido la metodología en cascada, donde las etapas se empiezan una detrás de la otra. Primeramente, se ha iniciado el proceso con la selección de lenguaje y tipo de programación, siguiendo por un análisis de las tareas a realizar, la gestión económica, el diseño, el desarrollo y finalmente la validación. La GUI ha sido desarrollada con lenguaje Python utilizando la programación orientada a objetos y librerías como PyQt5 para el diseño de la interfaz, Scikit-image para el procesamiento de imágenes y matplotlib para la representación e interacción entre el usuario y las imágenes.

El software permite al usuario seleccionar la imagen que desee, elegir entre diferentes procesamientos y segmentaciones para aplicar a la imagen. En caso de hacer una segmentación nos permite visualizar las características de cada región en una mesa, donde se relacionará cada etiqueta con sus propiedades. También es posible visualizar las propiedades de cada región en gráficos, de manera que las propiedades se compararan y se encontrar discordancias entre ellas.

Abstract

This project is a software development for pathological anatomy, with the aim of providing a tool for the visualization, processing and analysis of biomedical images in research centers for rare diseases. The purpose of the graphic user interface (GUI) that has been designed is the patterns research in diseased tissues, to gain knowledge about rare diseases detecting patterns in images.

For the project planning, the cascade methodology has been followed, where the stages starts one after the other. Firstly, the process has started with the selection of language and type of programming, followed by an analysis of the tasks to be performed, economic management, design, development and finally validation. The GUI has been developed with Python language using object-oriented programming (OOP) and libraries such as PyQt5 used for the interface design, Scikit-image used for image processing and matplotlib used for the representation and interaction between the user and the images.

The software allows the user selecting desired images, choose between different processing and segmentation to apply to the images. In case of doing a segmentation, it allows us to visualize the characteristics of each region in a table, where each label will be related to its properties. It is also possible to visualize the properties of each region in a graph, so that the properties are compared and disagreements are found between them.

Agraïments

En primer lloc m'agradaria agrair a en Raúl, director i mentor d'aquest projecte per confiar amb mi en tot moment, per donar-me l'ajuda necessària cada setmana i una injecció d'energia positiva per a no defallir en els moments difícils.

En segon lloc, a la meva família per no deixar de creure ni un moment en mi, per entendre les situacions d'estrès i desesperació que han sorgit mentre he après a programar. Els inicis sempre són difícils. I, tu, ja saps qui ets tu, amb els "tot anirà bé", per molts "no, si no ho faig millor, no hi anirà". I per últim a la Daniela, per guiar-me i fer-me entendre com pensa un programador.

A tots vosaltres i als que m'aprecieu, gràcies per estar-hi sempre.

Glossari

CREB o CREB-UPC: Centre de Recerca d'Enginyeria Biomèdica de la Universitat Politècnica de Catalunya.

TFG: Treball de final de grau.

UML: llenguatge unificat de modelització.

Python: llenguatge de programació.

PyQt: binding de la biblioteca gràfica Qt de C++ per a Python.

Skimage: scikit-image, llibreria per a processament d'imatge en Python.

Matplotlib: llibreria de Python per a dibuixar imatges i gràfics de qualitat en 2D.

Seaborn: llibreria de Python per a la representació de dades estadístiques amb estils diferents.

Script: arxiu d'ordres escrites en un llenguatge de programació.

OOP: Programació orientada a objectes.

Frameworks: estructura base per a l'organització i desenvolupament de programari.

V&V: Verificació i validació.

Plugins: aplicació que es relaciona amb una altra amb l'objectiu d'aportar-li noves funcionalitats.

csv: format d'un arxiu de taules on els valors estan separats per coma.

jpg: tipus de format gràfic basat en un algoritme de compressió d'imatges.

png: tipus de format gràfic basat en un algoritme de compressió d'imatges.

tif: tipus de format gràfic basat en un algoritme de compressió d'imatges.

Índex

1. Introducció

1.1 Origen del treball

El CREB-UPC, Centre de Recerca d'Enginyeria Biomèdica pertanyent a la Universitat Politècnica de Catalunya, té diverses divisions i una d'elles és la d'informàtica gràfica on es tracten les imatges biomèdiques i en la qual s'ha desenvolupat aquest treball de final de grau (TFG). Es va iniciar amb l'objectiu de col·laborar amb l'Hospital Sant Joan de Déu, proporcionant una plataforma que servís com a eina de suport per l'estudi de malalties rares i pogués ser utilitzada en els ordinadors de l'hospital.

1.2 Motivació

Les malalties rares o minoritàries com el segon nom indica, són aquelles malalties que tenen una baixa incidència en la població. A Europa són considerades aquelles que es presenten en menys de 5 persones de cada 10.000, tal que a Europa hi ha 27 milions de persones afectades i a Espanya 3 milions. A més a més, per si la baixa prevalença de les malalties rares no fos suficient, segons l'OMS, existeixen 7.000 malalties rares diferents que afecten un 7% de la població mundial (FEDER, 2019). La major part d'aquestes malalties són congènites, de patologia greu, cròniques, complexes i discapacitats que mai s'han arribat a diagnosticar o que ni tan sols es coneixen.

Per a fer recerca de les malalties rares és necessari un bon finançament, ja sigui per l'estat, les empreses farmacèutiques o d'ajudes voluntàries mogudes per projectes solidaris. La primera font d'ingressos anomenada no inverteix la suficient quantitat per arribar a subvencionar la recerca en aquest camp, ja que el Sistema de Salut Nacional té com a prioritat les malalties més freqüents en la població. D'igual forma les empreses farmacèutiques no inverteixen en la investigació i fabricació de medicaments per a una part tan petita de la població, ja que els hi suposaria moltes pèrdues i poques vendes. Per últim no són molts els projectes solidaris per a les malalties minoritàries, tot i que cal destacar que la Marató de TV3 aquest any, 2019, dedicarà la campanya a les malalties minoritàries (Corporació Catalana de Mitjans Audiovisuals, 2019) i FEDER entitat pública que lluita pels drets i l'adaptació dels malalts, promovent projectes solidaris entre les associacions afiliades.

En aquest treball s'ha volgut col·laborar en la recerca de malalties, on l'impuls inicial recau en les malalties rares, de poc interès per gran part de la societat actual, i per tant, poc estudiades. Des d'aquest projecte s'ha intentat desenvolupar una eina de processament d'imatge per a poder detectar patrons de diagnòstic en malalties d'anatomia patològica. Amb la finalitat de poder implementar la GUI en els equips dels hospitals o centres de recerca, per tal ajudant a extreure les dades rellevant que en un futur podrien arribar a definir el diagnòstic d'alguna malaltia rara.

1.3 Requeriments previs

Per a desenvolupar aquest projecte ha estat necessària la utilització d'un ordinador on fossin instal·lats Python, llibreries com ara numpy, sys, os i pandas, un entorn intel·ligent de desenvolupament, llibreries especifiques per a complir les funcionalitats requerides pels usuaris. Aquestes llibreries són **scikit-image** per al tractament d'imatges, **matplotlib** per a la visualització, interacció i extracció de dades estadístiques d'imatges, **seaborn** en la disposició de dades en gràfics i **PyQt5** per al disseny de la GUI.

Per altra banda ha estat necessària la utilització de programes de software per a la memòria, com ara Word per a la redacció i Draw.io per al disseny de diagrames UML, per a la planificació s'ha necessitat de TeamGantt una pàgina web per dissenyar diagrames de Gantt, Google Drive per el control de versions i servei de *backup* en cas de pèrdua de l'ordinador.

En últim lloc, però no menys important, ha estat necessària la comunicació entre l'Hospital Sant Joan de Déu i el director del projecte, la qual ens ha permès dissenyar un model de requeriment de la plataforma, on es defineixen les funcionalitats, l'aspecte i les limitacions.

1.4 Objectius i abast del treball

L'objectiu principal d'aquest projecte és la creació d'una interfície gràfica per a usuaris, que serveixi com a eina de suport en les imatges d'anatomia patològica, amb la finalitat de donar autonomia als centres de recerca de malalties rares. L'aplicació permetrà introduir, processar, visualitzar i extreure biomarcadors que ajudaran a definir els patrons de malalties desconegudes.

Per implementar aquesta millora general es defineixen els següents objectius específics:

- Disminuir la transferència de dades i reduir el perill d'agredir la identitat del pacient i poder mantenir l'anonimat.
- Oferir un camp de treball per a introduir imatges i poder enregistrar els resultats d'una manera ordenada, robusta, reproduïble, quantitativa i automàtica.
- Oferir la possibilitat d'introduir noves funcionalitats en el programa modificant el codi, com ara processaments, segmentacions, taules, gràfics i l'aspecte de la interfície.
- Oferir l'opció de multi processament, on l'usuari pugui escollir i comprovar quin es el que més s'adiu a les seves necessitats.
- Permetre l'extracció d'atributs i biomarcadors de rellevància clínica i funcional.

El principal objectiu és el processament i anàlisi d'imatges mitjançant una interfície senzilla i intuïtiva que permeti incorporar processaments personalitzables per resoldre problemàtiques concretes. Per altra banda, l'usuari ha de poder seleccionar la imatge que desitgi i aplicar-li el tipus de processat que més convingui, i no limitar el programa a un únic processament que pugui donar resultats poc satisfactoris.

La visualització paral·lela de la imatge original i la imatge segmentada o processada per a poder revisar si la segmentació ha estat ben realitzada i extreure diagnòstics més concloents.

L'extracció d'atributs i biomarcadors dels segments per a observar quines característiques té cadascuna de les regions, trobar semblances i diferències entre els controls i les mostres, de tal forma que aquelles diferències seran anotades i valorades per estudiader-les en més profunditat. Per representar les dades d'una forma més intuïtiva, s'afegirà el suport de gràfics on tota variació serà destacada.

L'objectiu general del projecte és participar en la investigació de diferents malalties desconegudes, proporcionant l'eina de treball que s'ha descrit en els objectius específics. A causa de la variabilitat entre malalties a estudiar, el programa té com a prioritat ser adaptable a les condicions de cada malaltia mitjançant la interacció entre programador i investigador.

2 Descripció del problema

2.1 Marc teòric

El marc teòric d'aquest projecte està orientat al desenvolupament d'una interfície gràfica d'usuaris (GUI) conjuntament amb el processament d'imatges biomèdiques com a funcionalitat principal de l'aplicació. Com a conseqüència s'explicaran els següents conceptes, GUI, llenguatge de programació orientat a objectes, UML, diagrames de classe, llibreries de processament d'imatges i d'interfície gràfica d'usuaris.

2.1.1 Interfície gràfica d'usuaris o GUI

Fins als anys 70, amb l'aparició de la computadora Xerox Alto, no s'havia sentit mai a parlar d'una interfície gràfica d'usuaris, ja que tots els usuaris es comunicaven amb l'ordinador mitjançant interfícies de línia d'ordres utilitzant un llenguatge específic, de manera que només els experts en podien fer ús. L'arribada de les GUI, una interfície que permet als usuaris entendre mitjançant relacions visuals com ara icones, finestres, botons, menús, entre d'altres i realitzar accions d'una manera més intuïtiva, ràpida i senzilla (García, 2015), permet la introducció d'usuaris no experts en el món dels ordinadors.

2.1.2 Programació i metodologia orientada a objectes

El desenvolupament d'un software engloba una sèrie de metodologies, tècniques i eines que faciliten les fases d'un projecte informàtic. Des dels anys 70, l'inici de l'època de la programació, quan s'estudia la millor forma d'organitzar aquest tipus de projecte. Malgrat que en aquells temps l'únic objectiu era optimitzar la velocitat de compilació i l'ús de la memòria, a mitjans de dècada va començar a aparèixer metodologies per a disminuir riscos i augmentar eficiència en la programació, fonamentalment el disseny estructural, i ja cap als anys 80 apareix la programació orientada a objectes (OOP).

Aquest projecte ha seguit la metodologia orientada a objectes, ja que ens aporta uniformitat, comprensió dels objectes que s'estan utilitzant, dels atributs i dels mètodes descrits en cadascuna de les classes. Aporta flexibilitat en les modificacions de les dades, ja que qualsevol canvi que afecti l'objecte quedarà reflectit en totes les funcions dins de la classe. El codi pot ser reutilitzat per programes que tinguin una estructura similar de dades i per tant, podrà servir per a desenvolupar softwares de processament d'imatge per a altres estudiants del grau. A més a més, seguint aquesta línia els programes són fàcilment llegibles i interpretables, deixant visible només la informació base dels programes, les classes. L'OOP divideix el problema en parts més petites que poden ser

comprovades de manera independent deixant l'amplitud de l'error limitada a seccions més petites i ràpidament identificables (Tejerina, 2011).

Per a representar aquesta metodologia s'utilitza el llenguatge unificat de modelització (UML), útil en la fase d'anàlisi i disseny on es comença a crear el model conceptual de dades que es farà servir. La creació de diagrames de classes dona més importància als elements del sistema des d'un punt de vista estructural i permanent. Així i tot no perden rellevància les relacions que hi ha entre les classes. A continuació se'n mostra la representació dels elements de l'UML en el diagrama de classes.

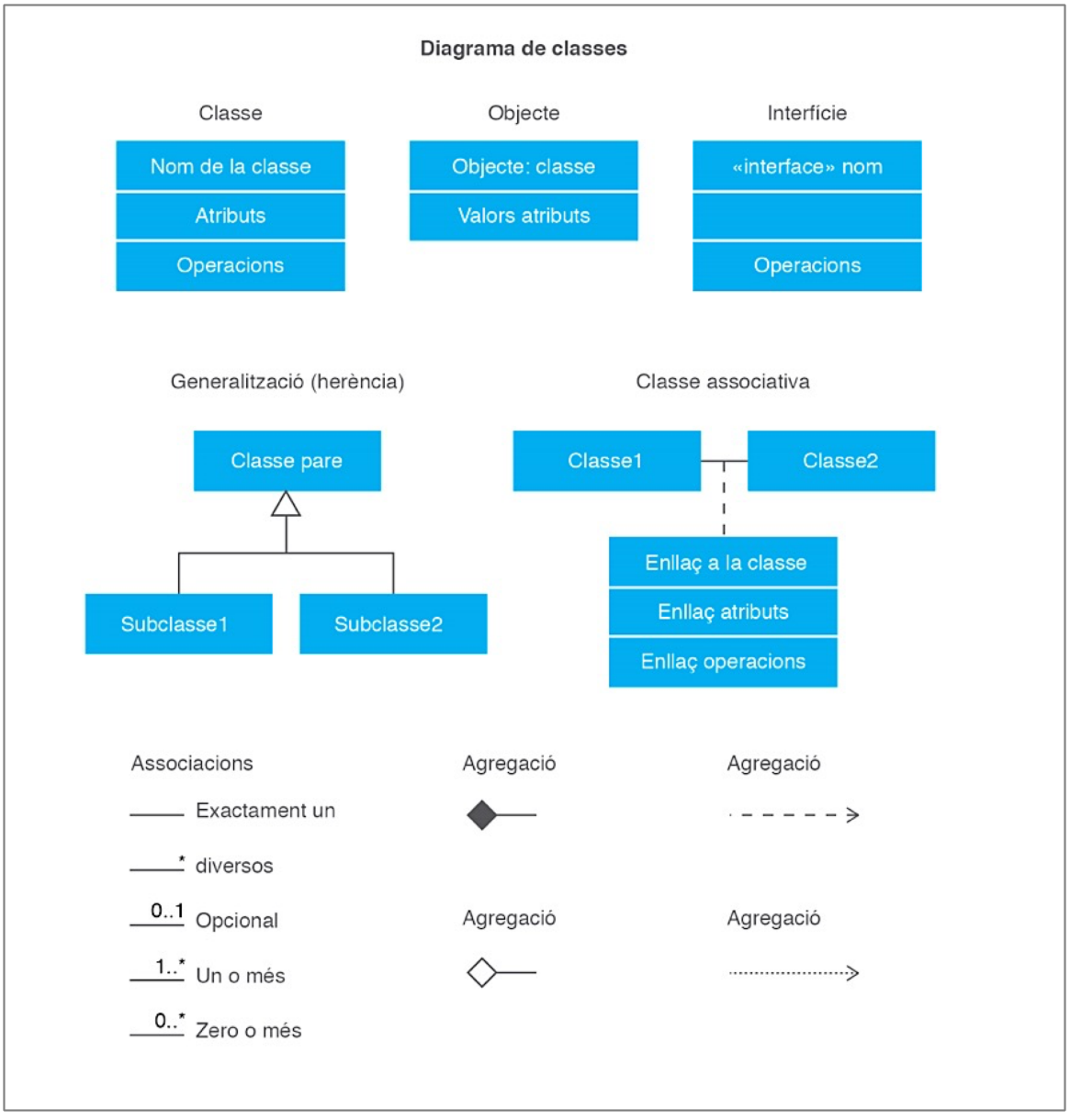


Fig 1: Elements del diagrama de classes. Font: GARCIA, M in line.

Les **classes** representen un conjunt d'objectes que comparteixen els mateixos **atributs**, característiques estables entre els objectes, i **operacions**, que representen les accions que es poden dur a terme sobre els atributs, produint canvis, obrir-los, tancar-los, carregar-los, etc. Els atributs i les operacions tenen 4 tipus de **visibilitats** diferents, *public* per a tots els elements del sistema, *private* només accessible per elements dels mateixos objectes, *protected* només és visible per elements del mateix objecte o especialitzacions, finalment la visibilitat *package* aplicable quan l'objecte no és un paquet i és visible per elements del paquet que conté l'objecte. L'últim element imprescindible en el projecte són les **relacions** o **agregacions** existents entre objectes que permeten cridar a un objecte 1 les operacions definides en un objecte 2 (García, 2012).

2.1.3 Llenguatge de programació

El llenguatge de programació és aquell que permet donar instruccions al maquinari en el qual es volen executar les ordres. Hi ha una gran quantitat de llenguatges, cadascun amb les seves pròpies característiques, especialitzacions, nivells, continguts, utilitats i estructures, de manera que requereixen un aprenentatge i una escriptura seguint rigorosament les regles sintàctiques i semàntiques. Aquestes regles permeten definir variables, classes, funcions i mètodes que defineixen les característiques i accions del programa.

Per aquest motiu s'ha fet una recerca dels diferents llenguatges de programació existents, per observar els avantatges i desavantatges que poden aportar a la plataforma a dissenyar. En primer lloc és necessari que aquest llenguatge sigui orientat a objectes, permeti el disseny d'una interfície gràfica on l'usuari pugui interaccionar amb les dades i sigui possible el tractament d'imatges amb processats típics per a la imatge biomèdica.

Durant la recerca s'ha observat que els que millor encaixaven per a dissenyar interfícies gràfiques eren C++, JavaScript i Python, i per al processament d'imatges C++, Python i Matlab, deixant endarrere l'opció d'utilitzar JavaScript. C++ és una de les opcions amb més prestigi, ja que és un llenguatge compilador, és a dir, el codi font és traduït al codi del maquinari mitjançant compiladors i no amb intèrprets que només tradueixen la informació quan és necessari (causant una baixa velocitat d'execució).

El llenguatge C++ és un llenguatge de programació d'alt nivell que requereix una implementació molt gran de temps per aprendre a programar, no té capacitat de ser utilitzat en multiplataforma directa, és a dir, necessita el desenvolupament d'un programa per a cada sistema operatiu. L'escriptura de codi i la utilització de dades són complexes, necessita declaració de cada variable, causant l'ampliació de l'extensió codi. Ja que el límit de temps del desenvolupament del projecte és un factor primordial, i el temps d'aprenentatge que requereix el C++ és molt més elevat que els llenguatges de Matlab i Python, aquest ha estat descartat.

El llenguatge M de Matlab és molt conegut entre els enginyers pel processament d'imatges, ja que compta d'un Toolbox especialitzat amb algoritmes per al processament, anàlisis i visualització d'imatges. Matlab permet crear diferents tipus de GUI, però té opcions de disseny molt limitades. A més no és un llenguatge de programació orientat a objectes. Tenint en compte que per al desenvolupament de l'aplicació és necessària una llicència de pagament i limitaria el nombre de possibles programadors que volguessin afegir un nou processat personalitzat. S'han investigat les llibreries matplotlib i scikit-image de Python, les quals estan especialitzades en el processament d'imatges a un nivell semblant a MATLAB. Python és un programari obert, multi plataforma i amb destinacions més generals per a la creació d'interfícies gràfiques intuïtives amb codis llegibles i reutilitzables per altres alumnes.

Per tant, Python ha estat el llenguatge escollit tenint en compte la seva senzillesa i rapidesa a l'hora de programar, és a dir, menys línies d'ordres per definir el mateix algoritme que en altres llenguatges. Un dels principals motius de la selecció d'aquest llenguatge han estat l'execució de funcions complexes de llibreries estàndards amb facilitat. L'existència de múltiples frameworks per al desenvolupament de les GUI com ara el binding PyQt5. De forma que programar en aquest llenguatge ens permet obtenir un codi net i transferible a altres programadors per a poder continuar aquest projecte.

2.1.4 Llibreries informàtiques

Per a facilitar el desenvolupament de codi existeixen les llibreries informàtiques o llibreries, subprogrames desenvolupats per altres programadors, on es defineixen funcions i mètodes que posteriorment són utilitzats per altres programes. En el projecte s'han utilitzat NumPy i Pandes, llibreries ja conegudes per la majoria i Scikit-Image, Matplotlib, Seaborn i PyQt5, llibreries especialitzades en processament i representacions d'imatges, gràfics i disseny d'interfícies gràfiques.

En primer lloc s'ha hagut de seleccionar la llibreria que més s'adaptés per complir els requisits de l'usuari en l'anàlisi i processament d'imatges. Durant la recerca s'han trobat dues llibreries les característiques que demanàvem OpenCV i Scikit-Image. La decisió no ha estat senzilla, ja que totes dues llibreries permeten el processament d'imatge a un nivell semblant, però finalment s'ha decidit utilitzar Scikit-Image el qual té una potència més elevada que OpenCV i la documentació és més completa, permetent un aprenentatge i una represa del projecte per altres estudiants més ràpida. És important destacar que en cas de necessitar alguna característica d'OpenCV que Skimage no permetés es pot instal·lar la llibreria i utilitzar-la també, ja que les dues llibreries treballen amb imatges de format np.ndarray.

Per altra banda Matplotlib és la llibreria que ens permet generar i visualitzar gràfics de la manera més coneguda per tots els estudiants, ja que és descendent de Matlab, el programari que es fa servir per a processament d'imatges. Per altra banda Seaborn ens aporta la facilitat de dissenyar gràfics amb

diferents estils i visualitzar-los amb matlabplot. I per últim s'ha investigat quines llibreries utilitzar per al disseny de GUI, de tal forma que s'ha decidit quin tipus d'aplicació es crearia, una aplicació d'escriptori. Les llibreries per a disseny web (Django , Flask o Pyramid) són descartades, per tant passem a les llibreries per a crear aplicacions d'escriptori. PyQt és un binding, una adaptació de la llibreria gràfica Qt desenvolupada en C++ per a Python. Destaca per ser estable, madura, multiplataforma i ser un completament natiu, és a dir que la adaptació al disseny del sistema operatiu és immediat.

3 Metodologia

La metodologia seguida en aquest treball té una gran semblança amb el cicle de vida d'un desenvolupament de software. El qual comença per unes necessitats, requeriments i objectius a assolir, continuant per la planificació, el disseny, la implementació, la validació, i finalment la publicació i el manteniment.

3.1 Estat de l'art i antecedents

Actualment el software obert més conegut que permet a l'usuari interaccionar amb la imatge, aplicar diferents filtres, processats, segmentats i obtenir paràmetres és ImageJ. El software ha estat propulsat i desenvolupat fonamentalment per Wayne Rasband en Java, el qual permet processar imatges en lots, efectuant els processats en paral·lel en hardwares multi CPU, permet interaccionar amb la imatge calculant distàncies, angles, àrees, valors de píxel, histogrames de densitat, gràfics de perfil de línia, processament d'imatges estàndards i transformacions geomètriques. A més a més, permet a l'usuari desenvolupar processats a mida utilitzant l'editor integrat i el compilador Java que posteriorment poden ser instal·lats com a *plugins* (Anònim, 2012).

No obstant això, ImageJ és un software poc intuïtiu per als investigadors especialitzats en àmbits diferents al processament d'imatge, ja que per a poder entendre quins són els paràmetres que s'han de modificar per obtenir el processat desitjat, és requereix d'una formació per a la utilització del programa. La problemàtica que es planteja en aquest projecte podria haver estat resolta amb el desenvolupament de *plugins* que incorporessin en el programa la funcionalitat de nous processats per als diferents tipus d'usuari als quals s'adreça aquest projecte.

Un aspecte molt important a tenir en compte és a qui va dirigit el software, i com volem que es mantingui i segueixi avançant. Per aquests dos motius s'ha decidit desenvolupar un altre software que aportes senzillesa, fos intuïtiu, ordenat i adaptat a les necessitats de l'usuari model, en aquest cas als investigadors de l'àrea d'anatomia patològica, generalment no experts en la programació de processament d'imatge.

El llenguatge de desenvolupament més adaptat a les funcionalitats que s'han d'introduir en el programa ha estat Python gràcies a les llibreries de processament d'imatge i visualització que tenen una estructura semblant al llenguatge M de Matlab après en l'assignatura de processament d'imatge d'enginyeria biomèdica. Aquesta semblança aproparà als futurs desenvolupadors a la concepció del codi realitzat i a la futura millora d'aquest.

3.2 Metodologia en cascada

Després d'analitzar quins tipus de metodologies existeixen per al desenvolupament de software, el model en cascada ha estat l'opció que millor s'adapta al projecte, ja que altres metodologies requerien un continu feedback del client, el qual no era possible. S'ha fet una petita variació en el disseny el qual implica la introducció de 4 etapes iteratives per a realitzar 5 versions del projecte i així poder afegir tots els requisits que s'han acordat en l'etapa inicial.

El disseny de les quatre versions s'ha realitzat a la primera etapa de disseny de forma general, en la resta d'etapes s'ha anat modificant el primer, afegint funcionalitats que a l'inici no s'han vist necessàries. Cada versió té el disseny adaptat a les funcionalitats que s'han d'anar implementant al llarg del desenvolupament. En cas de no arribar a cobrir totes les funcionalitats especificades en la versió, aquestes passaran a formar part de les millores de la següent versió. Així s'implementen millores de codi sense haver de tornar a definir els requisits de l'usuari, que s'aniran complint per parts. En la primera versió s'abastiran els principals i en les següents la resta, segons l'ordre de prioritat, fins que no hi hagi més temps per desenvolupar, i quedaran com a futures millores.

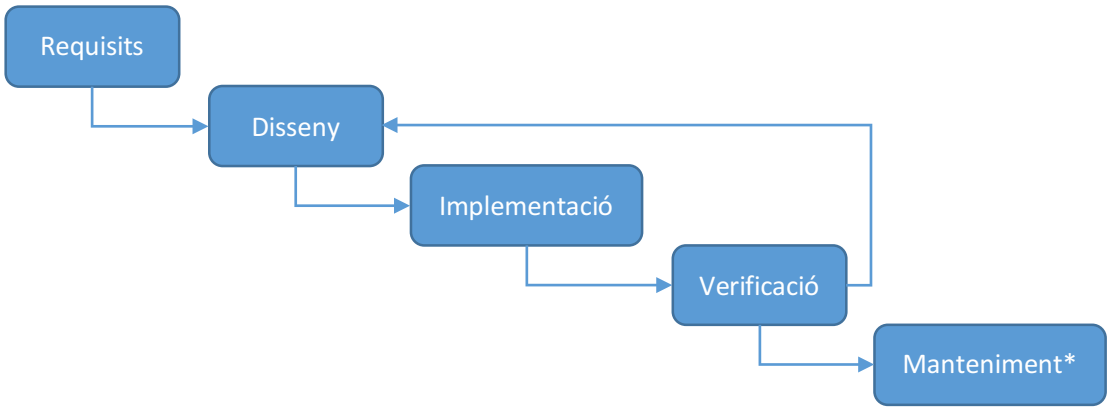


Fig 2: Variació del model en cascada utilitzat en el projecte.

*El manteniment no es realitza en el projecte, pensat per a futures funcions o implementacions.

Les etapes del model en cascada són les següents:

- **Requisits**: on s'analitzen els requisits que té el client per a definir les característiques i la funcionalitat que ha de complir el software a dissenyar, sense deixar-se'n cap, per no haver de tornar a repetir fases que ja s'han passat.
- **Disseny**: on es defineix quina és l'estructura interna del software i les relacions que s'han d'establir entre els components per a poder dur a terme els requisits establerts en la primera etapa.

- **Implementació**: on es desenvolupa el disseny de l'estructura de dades del software i de la interfície gràfica mitjançant el llenguatge de programació i les llibreries triades.
- **Verificació**: on s'analitza que el funcionament del software sigui correcte, sense tancaments del programa inesperats ni congelacions. Aquesta etapa porta inclosa una subetapa de correcció d'errors en la implementació.
- **Manteniment**: aquesta fase com indica la figura 2, no es realitzarà en el projecte, ja que requereix una implementació en l'hospital, un feedback dels usuaris i unes millores després de la posada en producció que no serà possible realitzar fins que el programa tingui implementat els processaments que interessen als usuaris.

3.3 Seguiment

Per dur un seguiment del projecte s'han fet reunions periòdiques, cada dilluns durant els 4 mesos de realització del projecte fins a l'entrega de la memòria. Cada reunió ha servit com a dada d'entrega de resultats de cadascuna de les fases i en cas de ser necessari, com a resolució de dubtes i replantejament de les possibles direccions del desenvolupament.

Per altra banda, també s'ha utilitzat Google Drive i fulls de càlcul per a portar un control dels canvis realitzats en cada versió i tenir present quines accions funcionaven i quines necessitaven ser millorades. On també s'ha anat pujant cada nova versió un cop passada per la fase de correcció fent servei de *backup*. De manera que, si es cometia un error greu en la modificació del codi, es podia revisar la versió anterior i corregir el problema, o en cas de no trobar l'error és partia de l'última versió per avançar el projecte.

3.4 Verificació i Validació

La verificació i la validació (V&V) són el conjunt de processos per a comprovar i analitzar el correcte funcionament del software, d'acord amb els requeriments de l'usuari i especificacions. A l'inici del desenvolupament d'un software pren un paper molt important, ja que qualsevol error en l'etapa principal va creixent exponencialment, arribant a provocar errors en les bases del programa que desemboquen a correcció de grans dimensions i una pèrdua important de temps.

La verificació i la validació tot i anar juntes no són el mateix, la verificació comprova que el programa es desenvolupi correctament i complint els requeriments de l'usuari, en canvi la validació comprova que el programa compleixi en termes generals les expectatives del client (López, 2017).

En aquest projecte s'ha dissenyat un procés de validació en el qual es farà una anàlisi de cada versió, tal que en cas de cometre un error greu la versió anterior serà el punt de partida i no passaran errors a les següents versions. Es comprovarà el funcionament del programa executant les versions i revisant totes les accions funcionin correctament. Se'n documentarà la validació que es farà a la versió de sortida del programa, el qual es presentarà en els annexos de la memòria.

4 Planificació

4.1 Planificació temporal

La planificació temporal del projecte està limitada pel període de temps des de l'assignació de TFG fins a l'entrega. Tenint en compte que aquest projecte abraça 24 crèdits ECTS i cada crèdit equival a 30 hores, en total s'hi dedica unes 720 hores repartides entre els mesos de febrer, març, abril ,maig i juny. Tot el projecte recau en una única persona amb cinc rols assignats, el de cap de projecte, enginyera de requisits, arquitecte de software, programadora i enginyera biomèdica especialitzada en el tractament d'imatges biomèdiques, amb l'ajuda del director del projecte per a recomanacions en la presa de decisions.

Les tasques considerades a l'inici del projecte amb les hores de dedicació estimades són les següents:

Tasca	Dedicació (tant per u)	Dedicació (h)
Elecció del llenguatge de programació	0,05	30,6
Estudi i proves de concepte	0,05	30,6
Gestió de projecte i requeriments d'usuari	0,1	61,2
Disseny de dades i arquitectura	0,1	61,2
Desenvolupament del programa	0,35	214,2
Control de qualitat i solució d'errors en el codi	0,1	61,2
Validació i Documentació	0,05	30,6
Memòria i presentació del projecte	0,2	122,4
Total	1	612*

Taula 1: Descripció de les tasques i hores dedicades.

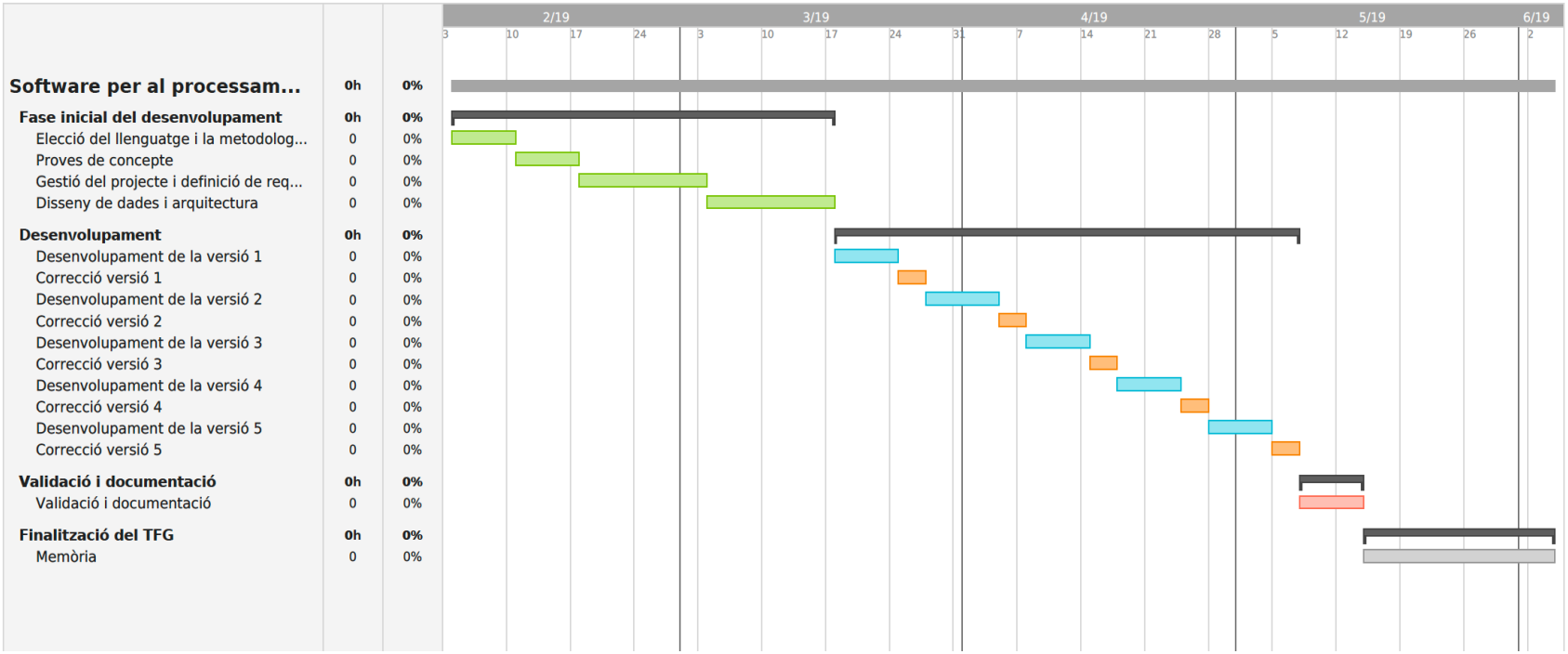
*Es considera una dedicació de 612 h, ja que les 108 h restants seran considerades com a desviacions i tenir en compte el risc d'un retard en el desenvolupament del projecte.

1. **Elecció de la metodologia i el llenguatge de programació**: ja que en el grau d'enginyeria biomèdica només es realitza una assignatura de computació, s'ha necessitat fer un estudi de les metodologies de software existents i els llenguatges de programació. Per a triar el mètode que més s'adapti al temps i situació del projecte, i el llenguatge que faciliti el desenvolupament d'una programadora no experta i així arribar a temps als terminis demanats.
2. **Proves de concepte**: en aquest apartat es comença a desenvolupar programes senzills per observar quines funcionalitats d'interfície gràfica podem arribar a realitzar amb la llibreria elegida (PyQt5), com ara el helloworld.py que es faria quan es comença a aprendre a programar. I en cas de no complir amb les expectatives, tenir temps de canviar de llenguatge.

3. **Gestió de projecte i requeriments d'usuari***:* en tot projecte es requereix una organització i definició de tasques en el calendari, ja que sempre hi ha terminis. Es necessita tenir present la càrrega de treball de cadascuna de les tasques a realitzar, per repartir-les en el temps de manera estratègica. Aquesta tasca afegeix també la càrrega de treball que comporta la definició dels requeriments de l'usuari abans de començar a dissenyar les funcionalitats del programa.
4. **Disseny de dades i arquitectura:** realització d'un disseny inicial del programa a un nivell bàsic, quines són les classes existents, amb els seus atributs i funcions que han de realitzar. Per altra banda, també es pensa quin és l'objectiu de les pantalles que s'han de crear, i per tant s'elabora un disseny esquemàtic de la interfície gràfica.
5. **Desenvolupament del programa:** un cop finalitzat el disseny es passa a escriure el codi per a la implementació de la GUI i l'estructura de dades.
6. **Control de qualitat i solució d'errors:** abans de donar per finalitzada una versió es passa per una fase de control de qualitat. En la que es comprovarà que no hi hagi errors en el funcionament del programa o en el disseny. Aquesta fase anirà acompanyada de la solució d'errors trobats en el codi fins arribar al resultat definit en la fase de disseny de la versió.
7. **Control de versions:** per evitar pèrdues de codis i tenir documentada cada versió amb els canvis més importants, s'afegeix la versió a Google Drive un cop revisada i es descriuen els canvis realitzat en la versió en la taula de control.
8. **Validació i Documentació:** un cop el programa es finalitza, es prepararà un test de validació que consistirà en l'execució del programa i la revisió de cada funcionalitat, registrant les evidències en forma de captures de pantalla. A més de la redacció de tota la documentació pertinent per a l'usuari i futurs programadors.
9. **Finalització del TFG:** finalitzar la redacció de la memòria del TFG, la preparació de la presentació i l'exposició davant del tribunal.

4.1.1 Diagrama de Gantt

A continuació es mostra la planificació temporal en un diagrama de Gantt on es reflecteix com s'ha pensat la repartició de les tasques en el temps per aconseguir la major eficiència i eficàcia del desenvolupament. Les tasques s'han agrupat en 4 fases, la fase inicial la qual introdueix les etapes d'elecció de llenguatge i metodologia, proves de concepte, gestió de projecte i disseny de dades i arquitectura del software. La segona fase és la del desenvolupament del software, on es troben les fases de desenvolupament de les versions i les seves corresponents correccions. La tercera fase només té una etapa de validació i verificació. I en quart lloc la fase de finalització del TFG on és redacta la memòria.



Taula 2: Diagrama Gantt per a l'organització del desenvolupament del software.

4.2 Planificació de hardware i software

El recurs de hardware utilitzat en aquest projecte és un ordinador MacBook Pro 2014 de 13 polzades amb un processador Intel Core i5 de 4 nuclis amb una velocitat de 2,6 GHz, una memòria RAM de 8 GB i sistema operatiu macOS Mojave. I els recursos de software Atom 1.36.1, Google Chrome 74.0.3729.131, Google Drive, Gmail, Draw.io, TeamGantt, Terminal 2.9.4, Anaconda 1.7.2 i Microsoft Office 2017.

Atom: és l'editor de codi que facilita suggeriments de lèxic i sintaxi pròpia del llenguatge de programació Python, permet navegar pels *scripts*, arxiu d'ordres escrites en llenguatge de programació, per projectes que agrupen tots els fitxers d'un mateix treball.

Google Chrome: com a navegador per a recerca d'informació i accés a pàgines web que s'utilitzen per a la redacció de la memòria, disseny de gràfics i diagrames, e-mail i emmagatzematge en el núvol.

Draw.io: pàgina web en línia per dissenyar diagrames UML.

TeamGantt: pàgina web en línia per dissenyar el diagrama de Gantt que representa l'organització del projecte.

Google Drive: per emmagatzemar les versions i portar el control en un full de càlcul en el núvol.

Gmail: el qual serveix com a mitjà de comunicació entre el director del projecte i l'autora. Per a planificar reunions, demanar dubtes, acordar disseny del programa i implementació de funcions si és necessari entre reunions.

Terminal i Anaconda: en la terminal s'executa el programa, en l'entorn de treball d'Anaconda, ja que allí estaran instal·lats els paquets necessaris.

Microsoft Office: editor de text per a redactar la memòria del projecte d'acord amb la plantilla de projectes de la universitat EEBE, i editor de càlcul per a fer taules amb pressupostos i per a la visualització esquemàtica d'alguns apartats de la memòria .

5 Model de requeriments

Per definir els requisits de l'usuari s'ha fet una reunió amb el director del projecte que estava en contacte amb investigadors de l'Hospital Sant Joan de Déu, els quals ens han servit com a usuaris de referència per adaptar-los a les seves necessitats i valorar quines implementacions serien les més adequades.

Els requeriments que demana l'usuari estan dividits en dos blocs, els generals i específics. Els primers són:

- Aplicació compatible amb sistema operatiu MacOs, Windows i Linux. Ja que la majoria d'ordinadors que utilitzen en els hospitals o centres de recerca públics tenen sistema operatiu Windows, al finalitzar el programa i ser validat amb un Mac s'avaluarà la compatibilitat amb el sistema Windows.
- Utilitzar Python com a llenguatge de programació per a una possible represa del projecte amb la implementació de noves funcions per altres estudiants i ampliació de l'abast del projecte. Ja que aquest llenguatge és fàcil d'entendre i implementar, la continuació del projecte podria ser represa amb facilitat i per tant arribar a tenir un manteniment del programa, que en cas de no haver-hi continuació no seria possible.
- El format d'imatges mèdiques que es processaran en el programa seran de formats jpg, png, tif, tot i que els formats de les imatges mèdiques originals són DICOM les imatges seran convertides externament a un format jpg per norma general.

I els requisits específics:

- Es demana poder visualitzar les imatges en l'aplicació i veure el resultat del processat en paral·lel, per poder comparar la imatge original amb la processada. D'aquesta forma l'usuari pot valorar si el processat que està utilitzant és l'adequat o per contra seria millor aplicar-n'hi un altre.
- Les imatges no tenen sempre la mateixa qualitat, tampoc es presenten les mateixes condicions ambientals ni de preparació de la mostra. Per tant, les imatges també necessiten diferents tipus de processat i segmentat. El programa ha de servir per diferents tipus d'imatges i per tan diferents tipus de segmentat segons els objectes a segmentar.
- Ja que és una eina per a la recerca, és important que tingui un apartat d'estudi estadístic i visual per a poder observar les característiques de les imatges en xifres, les quals sempre són més exactes que les imatges. També ha de poder visualitzar gràfics estadístics que permetran veure a simple vista desviacions entre els valors de diferents objectes.
- L'eina ha d'estar ben documentada, tant en el seu disseny com en les seves funcionalitats.

- L'eina ha de permetre la incorporació de tècniques de processament d'imatges desenvolupades per part de l'usuari per resoldre problemes concrets.

6 Disseny

El disseny del programa és la base del desenvolupament, en la que es busca trobar la millor manera de disposar les funcions del programa perquè sigui entenedor i intuïtiu per a l'usuari. S'ha dividit aquest apartat en tres blocs el disseny de la funcionalitat del programa, el disseny gràfic i el disseny de les classes del programa que seran utilitzades.

1. Casos d'ús

Per dissenyar una interfície gràfica és important tenir en compte quins casos d'ús ha de tenir el nostre programa, i així valorar la millor manera de mostrar a l'usuari els components amb els quals ha d'interaccionar. En el projecte s'ha desenvolupat un únic flux de casos d'ús, representat en la figura 3 mitjançant un diagrama UML dissenyat amb una plantilla d'UML Draw.io. L'usuari demana visualitzar una imatge, i pot processar-la o no, i segmentar-la. Un cop s'ha segmentat es poden extreure les propietats de cada segment, guardar les dades en un fitxer csv i finalment representar les dades en gràfics. A més les imatges i els gràfics es poden guardar en format d'imatge.

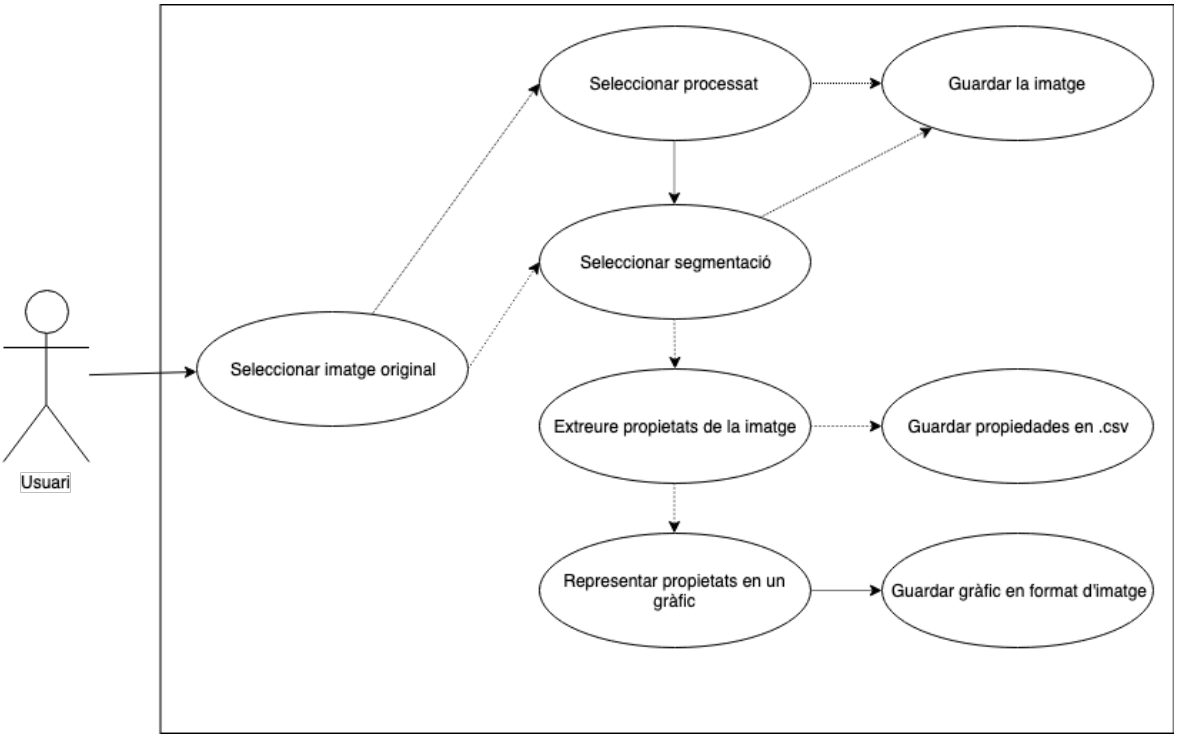


Fig 3: Diagrama UML dels casos d'ús del programa. Dissenyat amb: Draw.io.

6.1 Disseny gràfic

S'ha decidit crear 2 tipus de finestres, la primera finestra que troba l'usuari és la principal, aquella que permet fer totes les accions i la segona finestra és una derivada d'una acció ordenada en la primera. En la primera finestra es visualitzen les imatge i la taula amb les propietats, en canvi a la segona es visualitzen les propietats de la taula representades en gràfics.

En la Figura 4 hi ha la representació de l'últim disseny d'interfície gràfica que s'ha pensat per cobrir tots els requisits. En la part superior de la finestra s'afegeix una barra de menú amb 5 desplegables, el menú amb el nom de "Fitxer" ens serveix per inserir la imatge, el menú "Processat" ofereix l'oportunitat de triar el tipus de processat, el menú "Segmentació" permet triar el tipus de segmentació que es vol aplicar a la imatge i mostrar-ne el resultat, el menú "Taula" per a visualitzar els valors estadístics que ens donen informació de valor per a determinar la malaltia, el menú "Gràfic" ens permet fer gràfiques i el menú "Ajuda" on es pot visualitzar en una finestra de diàleg la informació del programa.

La part esquerra de la finestra principal ens permet visualitzar la imatge original i la imatge processada, de tal forma que es pot observar si el processat actua correctament o en cas contrari valorar l'opció d'aplicar-ne un altre. La barra d'eines que es presenta sobre la regió on es visualitzen les imatges és una eina de visualització per a poder ampliar, reduir, moure, definir els eixos i guardar les imatge.

En la part dreta, en canvi, hi ha una taula on s'indiquen les propietats quantificades de cada segment de la imatge. La qual disposa de dues barres de desplaçament per veure totes les columnes i files de la taula. A més disposa d'un botó per a guardar la taula en format CSV en la part inferior de la taula.

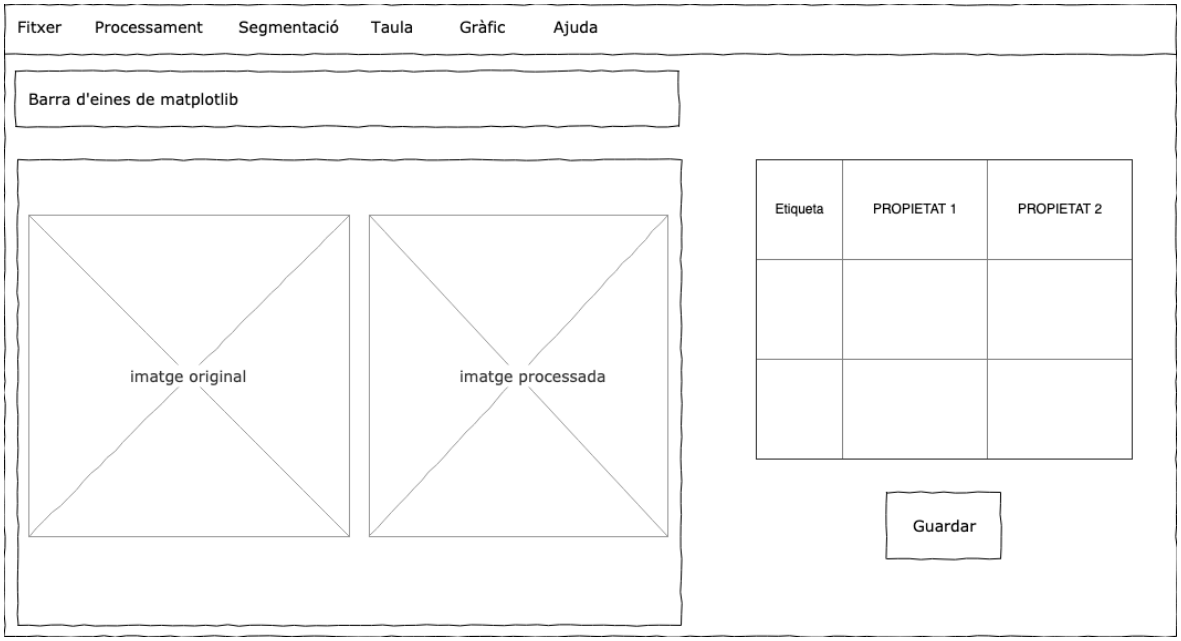


Fig 4: Disseny de la finestra principal del programa. Dissenyat amb: Draw.io.

En la Figura 5 que es mostra a continuació es representa la finestra que hauria de sortir en cas de seleccionar el menú “Gràfic”, i seleccionar el tipus de gràfic que es vol visualitzar. Com es pot observar també disposa d’una barra de navegació per a poder ampliar, reduir, moure els gràfics en cas de no veure correctament les dades i guardar los com una imatge. S’ha decidit el disseny d’una altra finestra, ja que la inclusió dels gràfics en la pantalla principal suposaria una visualització molt petita de cada component.

Fig 5: Finestra emergent per a la representació de gràfics. Dissenyat amb: Draw.io.

6.2 Estructura de classes

El diagrama UML de l’estructura de dades és molt útil per a tenir les funcions i mètodes de cada classe organitzats. S’ha dissenyat una estructura de dades general que permet afegir nous mètodes en cadascuna de les classes. En primer lloc, s’ha considerat una classe Scena que conté la imatge original, la imatge processada, la imatge segmentada i un DataFrame, estructura de dades típica de Pandas que guarda taules amb noms a cada columna, amb les propietats pertinents dels segments de la imatge. Aquests atributs s’extreuen mitjançant els mètodes representats en el diagrama UML de la figura 6.

A continuació es descriu la funcionalitat de cada mètode contingut en la classe Scena:

- **read_image**: encarregat d’extreure la imatge original seleccionada per l’usuari i el seu nom guardant los en els atribut images i nomimages.
- **gammacorrection:** aplica un processament de contrast gamma definit en la classe Segmentada i en guarda la imatge resultant a l’atribut process.
- **logcorrection**: aplica un processament de contrast logarítmic definit en la classe Segmentada i en guarda la imatge resultant a l’atribut process.

- **read_seg**: aplica la segmentació per llindar 0tsu definida en la classe Segmentada i en guarda la imatge resultant a l'atribut segmen.
- **watersheds**: aplica la segmentació per Watershed definida en la classe Segmentada i en guarda la imatge resultant a l'atribut segmen.
- **kmeans**: aplica la segmentació per K-menas definida en la classe Segmentada i en guarda la imatge resultant a l'atribut segmen.
- **felzens**: aplica la segmentació per felzenszwalb definida en la classe Segmentada i en guarda la imatge resultant a l'atribut segmen.
- **read_props**: llegeix les propietats de la imatge segmentada per Otsu i les guarda en l'atribut data.
- **read_propsw**: llegeix les propietats de la imatge segmentada per Watershed i les guarda en l'atribut data.
- **read_propsk**: llegeix les propietats de la imatge segmentada per K-means i les guarda en l'atribut data.
- **read_propsf**: llegeix les propietats de la imatge segmentada per felzenszwalb i les guarda en l'atribut data.

La classe Scena està relacionada amb la classe Segmentada per agregació, indicant que forma part de la classe Scena, però la destrucció de la classe contenidora no implicarà la desaparició de la classe continguda. La classe Segmentada té com a atributs la imatge original, el nom de la imatge, la imatge processada, la imatge segmentada i la imatge etiquetada. Els mètodes que conté són:

- **from_image**: el qual llegeix una imatge de format jpg, png o tiff i la converteix al format de numpy.ndarray guardant-la sota el nom de l'atribut imaori, i guardant el nom de la imatge extret del directori.
- **gammacorrection**: s'aplica el filtre de contrast gamma a la imatge imaori guardant el resultat en l'atribut process i crida al mètode reima.
- **logcorrection**: s'aplica el filtre de contrast logarítmic a la imatge imaori guardant el resultat en l'atribut process i crida al mètode reima.
- **reima**: guarda la imatge processada en imaori per a poder aplicar els filtres de segmentació a la imatge processada.
- **segmentacio**: segmenta la imatge per llindar Otsu. S'aplica un filtre de tancament previ a la segmentació, per a eliminar els possibles forats que hi puguin haver dins de cada segment i per eliminar els possibles artefactes afegits a les cantonades de l'objecte per el tancament, s'aplica un clear_border, el qual elimina els objectes afegits a les voreres del segment. Guarda la imatge segmentada en l'atribut label i la imatge amb la visualització dels segments en escala de color en l'atribut segmen.

- **segmentació_water**: aplica un filtre sobel a la imatge i la segmenta per Watershed guardant la imatge segmentada en l'atribut label i la imatge amb la visualització dels segments en escala de color en l'atribut segmen.
- **segmentació_k**: segmenta la imatge per k-means guardant la imatge segmentada en l'atribut label i la imatge amb la visualització dels segments en escala de color en l'atribut segmen.
- **segmentació_felz**: segmenta la imatge per felzenszwalb guardant la imatge segmentada en l'atribut label i la imatge amb la visualització dels segments en escala de color en l'atribut segmen.

Finalment hi ha una classe anomenada RegionProps que es relaciona amb la classe Scena per agregació, indicant que forma part d'aquesta, tenint en compte que la destrucció de la classe contenidora no implicarà la destrucció de la classe continguda. Per altra banda, RegionProps manté una relació de composició amb la classe Segmentada, és a dir, si la classe Segmentada desapareix, RegionProps ho fa a la vegada. Els seus atribut són una taula de propietats anomenada data del tipus Dataframe, llistes de les àrees, centroide i etiquetes de cada segment. Els mètodes que es troben en RegionProps són:

- **regseg**: a partir de la imatge segmentada per llindar Otsu s'extreu l'àrea, el centroide, la intensitat i l'etiqueta per després guardar-los en l'atribut data en format DataFrame.
- **regwat:** a partir de la imatge segmentada per Watershed s'extreu l'àrea, el centroide, la intensitat i l'etiqueta per després guardar-los en l'atribut data en format DataFrame.
- **regkmeans:** a partir de la imatge segmentada per K-means s'extreu l'àrea, el centroide, la intensitat i l'etiqueta per després guardar-los en l'atribut data en format DataFrame.
- **regfelz:** a partir de la imatge segmentada per Felzenszwalb s'extreu l'àrea, el centroide, la intensitat i l'etiqueta per després guardar-los com l'atribut data en format DataFrame.

Per tant, si es volen afegir noves tècniques de segmentació de la imatge, es dissenyen i s'afegeixen dins la classe Segmentada com a nous mètodes. Després es cridaran els resultats en la classe RegionProps i s'extrauran els atributs per després incloure la imatge segmentada i la taula amb les propietats de cada segment en la classe Scena. En cas de voler afegir processats, únicament es dissenyaran els nous mètodes contenidors dels processament i s'implementaran en la classe Scena, cridant el mètode reima que convertirà la imatge processada en original, permeten segmentar la imatge processada en comptes de l'original.

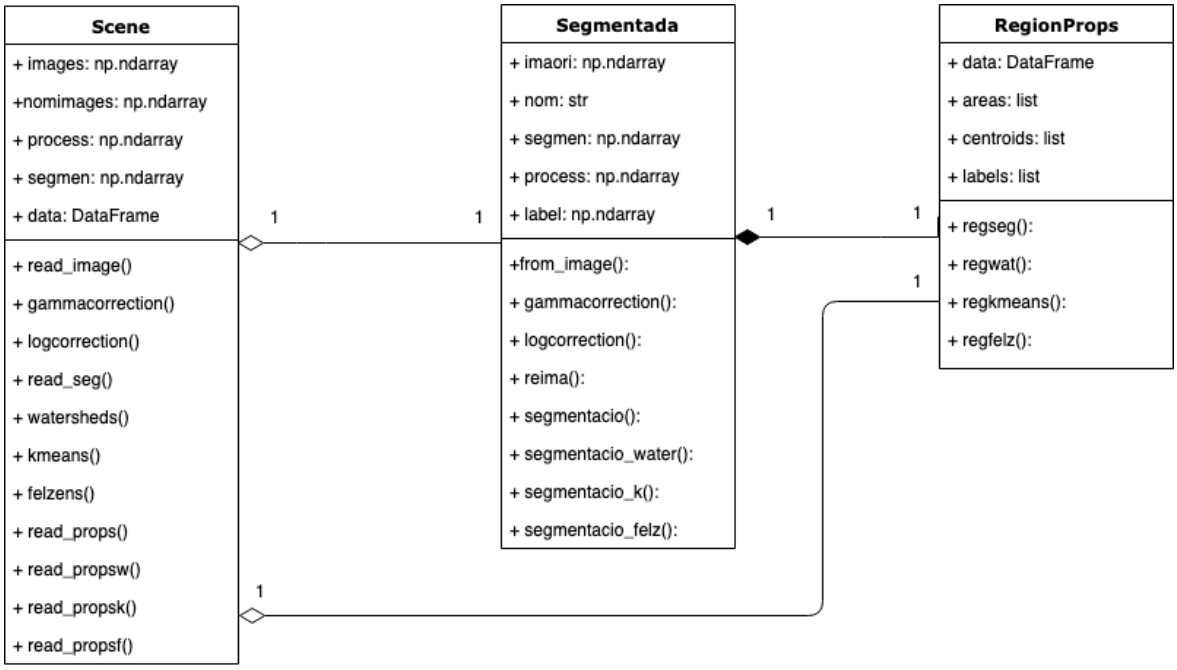


Fig 6: UML de l'estructura classes del programa. Dissenyat amb: Draw.io

7 Desenvolupament

El programa consisteix en un conjunt d'*scrips* diferenciats per la seva funcionalitat, on es defineix l'estructura de dades o el disseny de l'aplicació. En termes generals, l'aplicació comença amb un script de disseny anomenat *projecte.py* on és construeix la finestra principal del programa, amb la seva barra de menús, on es troben les opcions de selecció de la imatge, de processat, de segmentat, de taula i de gràfics les quals estan lligades a unes funcions que permeten visualitzar la informació en la finestra mitjançant un widget central, o en el cas del gràfic en una finestra emergent.

El widget central esta definit en una altre *script* anomenat *centralwidget.py* on s'estableix la disposició dels widgets en la pantalla, utilitzant els mètodes que s'expliquen posteriorment. En *plot_canvas.py* i en *table_widget.py* és defineix la disposició de les imatges amb el tipus d'eixos, quantitat d'imatges a visualitzar, la seva disposició i la taula de propietats de cada regió de la imatge segmentada. El fitxer pandesmodel extret de GitHub (Yllanes, 2018) dóna suport a table_widget per a dipositar les dades en format Dataframe (estructura elemental de pandes) a una taula Qt. I per últim lloc en els scripts de disseny tenim el *plot_scatter.py* on es defineix la visualització en una finestra emergent de matplotlib del gràfic scatter.

L'aplicació compta amb l'script *scene.py* que actua com a nexe entre les dades i el disseny gràfic, aquest emmagatzema la informació que incorpora l'usuari, la processa, segmenta i extreu les propietats de les regions segmentades mitjançant les funcions definides en les classes principals, segmentada.py i regionprops.py. En la classe segmentada es podrà extreure la imatge original, la imatge processada i finalment la imatge segmentada, en canvi en la classe regionprops s'obtindran els DataFrames amb les propietats de cada regió.

Un cop vista la descripció general del funcionament del programa es mostraran les funcions utilitzades a cada script i la descripció de les funcionalitats per enllaçar el motiu de la seva utilització.

7.1 Segmentada

Script	Segmentada		
Llibreries utilitzades	**Nom de la llibreria**	**Funció**	**Descripció de funcionalitat**
	Scikit-image	io	La funció io definida dins de skimage s'utilitza per a llegir les imatges, canviant el format jpg a format np.ndarray (una matriu de numpy)

		threshold_otsu	La funció threshold_otsu situada en el mòdul filters retorna el valor de llindar d'intensitat de la imatge basada en el mètode Otsu.
		sobel	La funció sobel situada en el mòdul filters detecta les vores dels objectes aplicar el filtre sobel.
		clear_border	La funció clear_borders situada en el mòdul segmentation elimina artefactes lligats a les voreres dels objectes trobats en la imatge segmentada
		watershed	La funció watershed situada en el mòdul segmentation segmenta la imatge amb la tècnica de transformació divisòria, en la qual s'interpreta la imatge en escala de grisos com una imatge topografica on el blanc es el punt més alt i el negre el més baix.
		felzenszwalb	La funció felzenswalb situada en el mòdul segmentation calcula la manera més eficient de segmentar una imatge basada en els gràfics de Felzenszwalb.
		slic	La funció slic situada en el mòdul segmentation permet segmentar les imatges a partir de la classificació per k-means.
		label	La funció label situada en el mòdul measures etiqueta les regions connectades entre si en una imatge en forma de matriu.
		closing	La funció closing situada en el mòdul morphology fa un tancament morfològic, una dilatació seguida d'una erosió, per tal d'esborrar el soroll anomenat pebre que hi pugui haver en la imatge.

		square	La funció square situada en el mòdul morphology crea una matriu quadrada que és fa convolucionar en la funció closing.
		resize	La funció resize, situada en el mòdul transform, redimensiona la imatge fent-la coincidir amb el valor que s'indica.
		label2rgb	La funció label2rgb en el mòdul color transforma una imatge d'etiquetes en una imatge a color per distingir els diferents elements.
		rgb2gray	La funció rgb2gray en el mòdul color transforma una imatge a color en una imatge a escala de grisos.

Taula 3: Funcions utilitzades en l'script Segmentada.py.

7.2 RegionProps

Script	RegionProps		
Llibreries utilitzades	**Nom de la llibreria**	**Funció**	**Descripció de funcionalitat**
	Pandas	DataFrame	La funció DataFrame de Pandas serveix per a crear l'estructura elemental de pandas, una estructura de dades bidimensional amb eixos etiquetats en files i columnes.
	Scikit-image	regionprops	La funció regionprops situada en el mòdul measures obté les propietats de cada regió segmentada i etiquetada.

Taula 4: Funcions utilitzades en l'script RegionProps.py

7.3 Scene

El nexe entre les dades i la interfície gràfica és l'script scene.py el qual no ha utilitzat cap llibreria per a definir-lo, únicament s'ha cridat als *scripts* segmentada.py i regionprops.py per fer ús dels seus mètodes o a la inversa ha estat cridat per l'*script* project .py.

7.4 Project

Script	project			
Llibreries utilitzades	**Nom de la llibreria**		**Mètodes**	**Descripció de funcionalitat**
	PyQt	Classe	QMainWindow	Defineix la classe de finestra principal de l'aplicació amb possibilitat d'afegir barra d'estat, barres d'eines i una barra de menús.
			QFileDialog	Proporciona una finestra del tipus diàleg que permet als usuaris seleccionar documents.
		Mètode	setWindowTitle	Defineix el nom a la finestra.
			setGeometry	Defineix la mida de la finestra, indicant en la posició x, la y, l'amplada i l'alçada de la finestra, en aquest ordre.
			setCentralWidget	Estableix el widget principal de la finestra.
			menuBar	Crea una barra de menú per a la finestra.
			addMenu	Afegeix un menú a la barra de menús.

			QAction	Defineix les accions realitzades mitjançant els components d'una barra de menús, una barra d'eines o amb una drecera de teclat personalitzada
			setShortcut	Defineix una drecera de teclat per a un component de la barra de menús
			addAction	Afegeix l'acció al menú
			setStatusTip	Afegeix un estat a la barra de menú, que es visualitza al mantenir el ratolí al damunt.
			statusBar	Crea una barra d'estat on es visualitza informació útil dels processos o accions que s'estan duent a terme en aquell moment.
			showMessage	Permet afegir la informació que es vol visualitzar a la barra d'estat.
			getOpenFileName	En les classes QFileDialog permet obrir la finestra de diàleg, determinant el tipus de fitxers que es poden obrir.
			QMessageBox	Crea un diàleg modal utilitzat per a mostrar algun missatge informatiu i opcionalment demanar a l'usuari interaccioni per afirmar o negar que vol seguir amb l'acció corresponent.
			Qapplication	Gestiona la configuració de l'aplicació i el flux de control de l'aplicació GUI.
	sys (mòdul del sistema)	mètode	argv	Crea una llista de tots els arguments de la línia d'ordres.

			exit	Força la sortida de l'intèrpret de Python.

Taula 5: Mètodes i classes utilitzades en project.py

7.5 CentralWidget

Script	centralwidget			
Llibreries utilitzades	**Nom de la llibreria**		**Mètodes**	**Descripció de funcionalitat**
	PyQt	Classe	QWidget	És la classe base per a la definició d'objectes les GUI de PyQt, el qual per defecte no té progenitors si no s'indica.
		Mètodes	QVBoxLayout	Gestiona la disposició dels widgets en la finestra de l'aplicació, dipositant-los verticalment.
			addWidget	Afegeix un widget en el disseny de la finestra.
			QHBoxLayout	Gestiona la disposició dels widgets en la finestra de l'aplicació, dipositant-los horitzontalment.
			addLayout	Afegeix el contenidors de widgets, BoxLayout, a altres BoxLayout.
			setLayout	Defineix el contenidor de widgets de la finestra.

			parent	Per accedir a la finestra principal a partir d'una altra classe de finestra.
			statusBar	Per afegir una barra d'estat a la finestra. Només es pot afegir a la finestra principal, per tant s'utilitza parent().statusBar()
			showMessage	Defineix el missatge que es mostrarà en la barra d'estat.

Taula 6:Mètodes i classes utilitzades en l'script centralwidget.py

7.6 Plot_canvas

Script	plot_canvas			
Llibreries utilitzades	**Nom de la llibreria**		**Mètodes**	**Descripció de funcionalitat**
	PyQt	classe	FigureCanvas	Classe que permet disposar figures de matplotlib en les finestres, conjuntament amb altres widgets.
			setParent	Defineix de qui és propietat el Qwidget
			setSizePolicy	Defineix la política de redimensionament de les mesures verticals i horitzontals de les finestres.

			updateGeometry	Actualitza les mesures de la finestra segons la política definida anteriorment.
	Matplotlib	functions	subplots	Crea subgràfics dins del visualitzador de matplotlib.
			set_tight_layout	Endreça els subgràfics de forma que es vegin tots els paràmetres, eixos, títols, etc.
			add_subplot	Afegeix subgràfics a la figura que s'està indicant.
			clear	Neteja la zona de cada subgràfic en el que s'indica que es netegi, deixant
			imshow	Mostra la imatge en els eixos que s'indiquen.
			draw	Dibuixa tot el que s'ha dissenyat en la pantalla, eixos, imatges, gràfics, llegendes, etc.

Taula 7:Mètodes i classes utilitzades en l'script plot_canvas.py

7.7 Table_widget

Script	table_widget			
Llibreries utilitzades	**Nom de la llibreria**		**Mètodes**	**Descripció de funcionalitat**
	PyQt	classe	QWidget	És la classe base per a la definició d'objectes les GUI de PyQt, el qual per defecte no té progenitors si no s'indica.

			QFileDialog	Proporciona una finestra del tipus diàleg que permet als usuaris seleccionar documents.
		Mètodes	setFixedSize	Fixa l'amplitud i l'alçada de les dimensions del widget.
			QTableView	Ens permet utilitzar un model de taula per visualitzar.
			QPushButton	Crear botó
			clicked.connect	Connectar el botó a una acció
			QVBoxLayout	Gestiona la disposició dels widgets en la finestra de l'aplicació, dipositant-los verticalment.
			addWidget	Afegeix un widget en el disseny de la finestra.
			setModel	Estableix el model per a la presentació de la vista.
			getSaveFileName	Guarda fitxers en el destí especificat amb la finestra QFileDialog.
	pandas	-	to_csv	Escriu un csv amb les dades introduïdes.

Taula 8: Mètodes i classes utilitzades en l'script table_widget.py

7.8 Plot_scatter

Script	plot_scatter			
Llibreries utilitzades	**Nom del llibreria**	**Mòdul**	**Funció**	**Descripció de funcionalitat**
	PyQt	classe	QWidget	És la classe base per a la definició d'objectes les GUI de PyQt, el qual per defecte no té progenitors si no s'indica.
	Matplotlib	pyplot	figure	Crea una nova figura de matplot
			canvas.set_window_title	Defineix el títol de la finestra que mostra el gràfic.
			subplot	Crea subgràfics dins del visualitzador de matplotlib.
			show	Mostra la figura matplot o els gràfics que s'han creat en una nova finestra de matplot
	Seaborn	sns	scatterplot	Crea un gràfic de punts.

Taula 9: Mètodes i classes utilitzades en l'script Segmentada.py

A part dels fitxers del programa s'ha creat un altre fitxer en format text anomenat README.txt que dóna les instruccions necessàries per a poder executar el programa destacant les condicions

necessàries per al funcionament d'aquest. A continuació s'adjunta la informació del fitxer de text readme.txt.

README.text

El programa que esteu a punt d'utilitzar requereix instal·lació prèvia de Python versió 3 en amunt, pip, eina d'instal·lació de paquets de Python i un conjunt de paquets (PyQt5, scikit-image, matplotlib, numpy, pandas, seaborn i mpldatacursor) els quals s'instal·laran executant la següent comanda en el terminal.

```
>>> pip install *
```

on * és el lloc on anirà el nom dels paquets que he anomenat anteriorment, i s'executarà d'un en un, és a dir:

```
>>> pip install numpy

>>> pip install pandas

>>> pip install PyQt5

>>> pip install scikit-image

>>> pip install matplotlib

>>> pip install seaborn

>>> pip install mpldatacursor
```

Un cop s'ha fet la instal·lació de totes les llibreries necessàries es procedirà a executar el programa amb la següent línia de codi:

```
>>>python ./project.py
```
(en MacOs)

```
>>>python project.py
```
(en Windows i Linux)

8 Funcionalitats

Les funcions que té el programa seran explicades en aquest apartat mitjançant una documentació de la guia per a l'usuari, amb la finalitat de descriure un manual d'ús i una altra per al programador, per a mostrar com modificar el codi per a aconseguir implementar noves funcionalitats en el programa.

8.1 Manual d'ús per a l'usuari

En primer lloc, l'usuari pot mostrar una imatge en la plataforma seguint els següents passos:

- Seleccionar **Fitxer** > **Seleccionar imatge** en la barra de menú.

Per processar i visualitzar la imatge amb els diferents tipus de processament existents en el programa:

- Un cop seleccionada i visualitzada la imatge es selecciona **Processament** > ***tipus de processat que desitgi l'usuari aplicar.***
  Actualment existeixen dos tipus de processats en l'aplicació dins del menú Processament: de correcció de contrast gamma sota el nom **Contrast gamma**, i de correcció de contrast logarítmic sota el nom **Contrast log**.

Per segmentar i visualitzar la imatge amb els diferents tipus de segmentació presents en l'aplicació:

- És necessària la selecció d'una imatge i és possible aplicar els processaments, però no obligatori.
- Seleccionar **Segmentació** > **tipus de segmentació que desitgi l'usuari aplicar.**
  Actualment existeixen quatre tipus de segmentació en l'aplicació dins del menú Segmentació: la segmentació per llindar Otsu sota el nom **Otsu**, la segmentació per Watershed sota el nom de **Watershed**, la segmentació per K-means sota el nom de **K-means**, la segmentació per Felzenszwalb sota el nom de **Felzenszwalb**.

Per a extreure les propietats de cada regió de la imatge segmentada en una taula:

- És necessària la selecció d'una imatge i si es desitja també es pot aplicar processaments però no es requereix segmentar la imatge per a visualitzar les propietats.
- Seleccionar **Taula** > **dades del tipus de segmentació que l'usuari vulgui obtenir**.
  Actualment existeixen tantes taules com segmentacions indicades en l'explicació anterior, **Propietats d'Otsu**, **Propietats de watershed**, **Propietats de k-means**, **Propietats de felzenszwalb**.

Per a visualitzar les dades de les taules en gràfics que aportin informació en una pantalla emergent:

- És necessària la selecció d'una imatge i un processament en cas de voler obtenir els valors de la imatge processada. No es necessita segmentar ni extreure les dades en la taula per a visualitzar el gràfic.
- Seleccionar **Gràfic** > **Area/Intensitat**.
  Actualment només s'ha afegit un tipus de gràfic però es poden dissenyar altres tipus i inserir-los en el menú Gràfic, tal i com s'explica en l'apartat de **Documentació per a futurs programador**.

Altres funcionalitats que té la interfície és el desament de les imatges, taules i gràfics. Per guardar les imatges de la pantalla principal i el gràfic:

- Seleccionar la **icona del disquet** > escriure el **nom del fitxer** > seleccionar l'**etiqueta** > seleccionar el **directori** en el qual es vol emmagatzemar > seleccionar el tipus de **format**.

Per a guardar la taula de dades en format csv, l'usuari ha de:

- Seleccionar el botó sota la taula de dades amb el nom de **Guardar** > escriure el **nom del fitxer** > seleccionar l'**etiqueta**> seleccionar el **directori** en el qual es vol emmagatzemar.

Per ampliar la imatge s'ha de fer servir la icona de la **lupa** o la de les **barres desplaçables** i per a moure els eixos la **creu amb puntes de fletxa**. Si es volen modificar els eixos, es pot fer servir la icona de la gràfica, deixant triar l'eix que es vol modificar i posteriorment els paràmetres a modificar.

Per obtenir la informació de l'aplicació cal:

- Seleccionar en el menú **Ajuda** > **Sobre el programa**. Que mostrarà una finestra de diàleg amb la informació.

8.2 Documentació per a futurs programadors

En aquest apartat es defineixen els passos que han de seguir els futurs programadors per a afegir nous processaments, segmentacions, taules i gràfics. Per implementar un nou processat:

- S'afegeix un nou mètode de processat en l'script **segmentada.py**.
- Es crea un nou mètode en l'script **scena.py** que emmagatzema la imatge processada en l'atribut *process* cridant al nou mètode de la Segmentada. I es crida al mètode reima per a definir la imatge processada com a imatge original.

- Es crea un nou mètode en l'script **project.py** que crida a l'atribut *process* de l'Scena i l'utilitza com a paràmetre del mètode render_seg, del central_widget. El mètode render_seg permetrà visualitzar la imatge en la pantalla principal.
- Es crea una nova opció en el menú Processament de **project.py**, amb el nom del processat i es connecta al nou mètode creat.

Per implementar una nova segmentació:

- S'afegeix un nou mètode de segmentació en l'script **segmentada.py**.
- Es crea un nou mètode en l'script **scena.py** que emmagatzema la imatge segmentada en l'atribut *segmen* cridant al nou mètode de la Segmentada.
- Es crea un nou mètode en l'script **project.py** que crida a l'atribut *segmen* de l'Scena i l'utilitza com a paràmetre del mètode *render_seg*, del *central_widget*. El mètode *render_seg* permetrà visualitzar la imatge en la pantalla principal.
- Es crea una nova opció en el menú Segmentació de **project.py**, amb el nom de la segmentació i es connecta al nou mètode creat.

Per implementar una nova taula d'acord amb la nova segmentació:

- S'afegeix un nou mètode per extreure les propietats de les regions, de la nova segmentació, en l'script **regionprops.py**.
- Es crea un nou mètode en l'script **scena.py** que emmagatzema les propietats de les regions en l'atribut *data* cridant al nou mètode del Regionprops.
- Es crea un nou mètode en l'script **project.py** que crida a l'atribut *data* de l'Scena i l'utilitza com a paràmetre del mètode *render_prop*, del *central_widget*. El mètode *render_prop* permetrà visualitzar les característiques de cada segment en una taula.
- Es crea una nova opció en el menú Taula de **project.py**, amb el nom de la segmentació utilitzada i es connecta al nou mètode creat.

Per implementar una nova gràfica d'acord amb la nova segmentació i/o amb les característiques de les regions que es volen visualitzar:

- Es crea un nou mètode en l'script **project.py** que crida a l'atribut *data* de l'Scena i l'utilitza com a paràmetre del mètode *render_scat*, del *plot_scatter*. El mètode *render_scat* permetrà visualitzar les característiques de cada segment en una gràfica.
- Es crea una nova opció en el menú Scatter de **project.py**, amb el nom del gràfic i els paràmetres a visualitzar i es connecta al nou mètode creat

9 Resultats i Validació del programa

Per avaluar els resultats del software desenvolupat es comproven totes les funcionalitats que hem creat i s'observen si les realitza correctament, o per contra necessita correccions que quedaran descrites en l'apartat de futures millores per aplicar en el programa. Per altra banda, aquest apartat també s'utilitzarà com a validació final del programa on es descriuran els testos que s'han creat per a comprovar el funcionament del software, i les pantalles resultants de les execucions dels testos. Cal tenir en compte que la fase de validació és present en totes les etapes del desenvolupament de software, tot i així només quedarà documentada l'última versió.

Per l'execució del programa, com ja s'ha dit en l'apartat de requeriments previs, hi ha una sèrie de prerequisits que s'han de complir, entre ells la instal·lació de Python, PyQt, scikit-image, matplotlib, seaborn, numpy i pandas en el ordinador. Un cop ens hem assegurat que els prerequisits es compleixen, passem a l'execució dels tests.

9.1 Test 1: Execució del programa

ID del test	T.01
Nom del test	Execució del programa.
Objectiu/Descripció del test	Demostrar la correcta execució del programa.
Criteri d'acceptació	El sistema permet executar el programa i visualitzar la interfície gràfica.

Taula 10: Descripció del test 1.

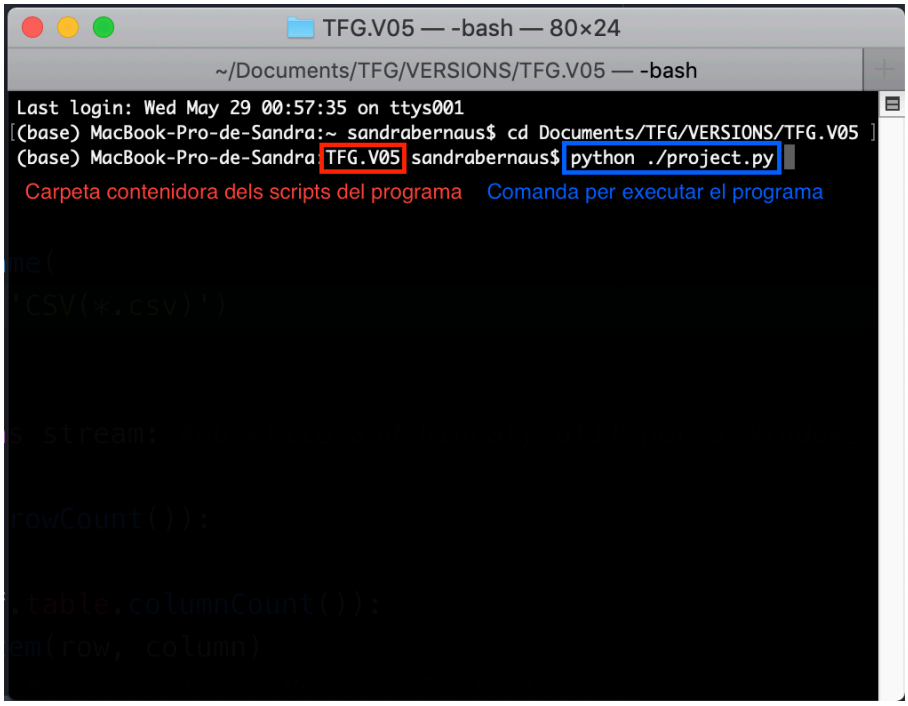


Fig 7: Línia de comandes per a l'execució del programa.

En la figura 7 es mostra les comandes que són necessàries per executar el programa des de la Terminal. Primerament entrem en el directori on es troba l'script projecte.py, i executem la comanda `python ./project.py` en cas de ser sistema operatiu MacOs, si es Windows o Linux s'hauria d'escriure la comanda `python project.py`.

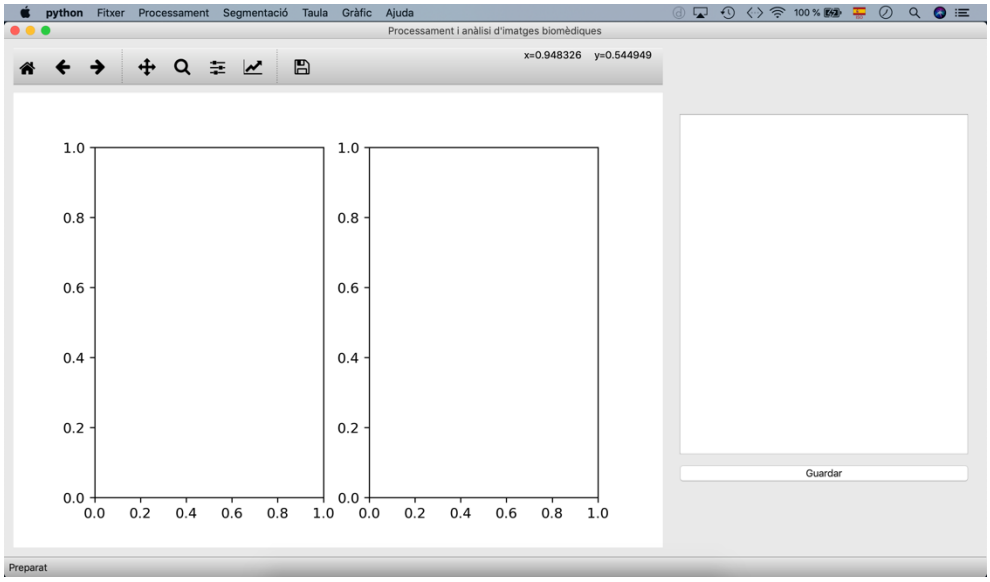


Fig 8: Finestra principal del programa.

En aquesta pantalla s'observa com s'obre el programa després d'haver executat la comanda anterior, i es pot comprovar com el programa s'estructura de la mateixa forma que s'ha dissenyat, presenta la barra de menús amb totes les opcions, la pantalla de visualització d'imatges amb la barra d'eines i l'espai on es visualitzarà la taula amb el botó per a guardar-la en csv.

9.2 Test 2: Visualització d'imatges

ID del test	T.02
Nom del test	Visualització d'imatges.
Objectiu/Descripció del test	Demostrar la possibilitat de seleccionar diferents tipus d'imatge mitjançant l'opció de 'Seleccionar imatge' del menú 'Arxiu' i visualitzar-les.
Criteri d'acceptació	El sistema permet seleccionar imatges de formats jpg, png i tiff i visualitzar-les correctament.

Taula 11: Descripció del test 2.

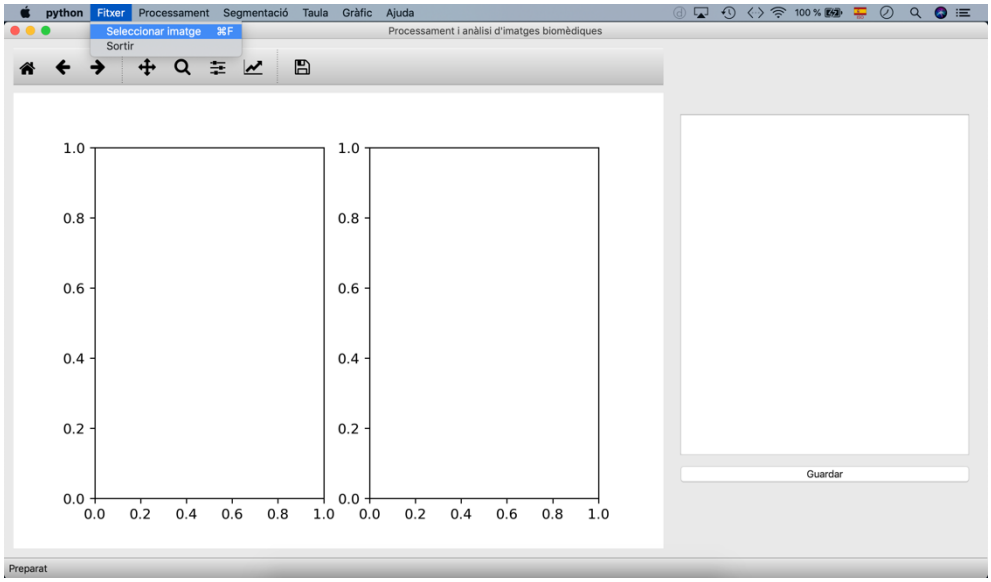


Fig 9: Acció per seleccionar fitxer.

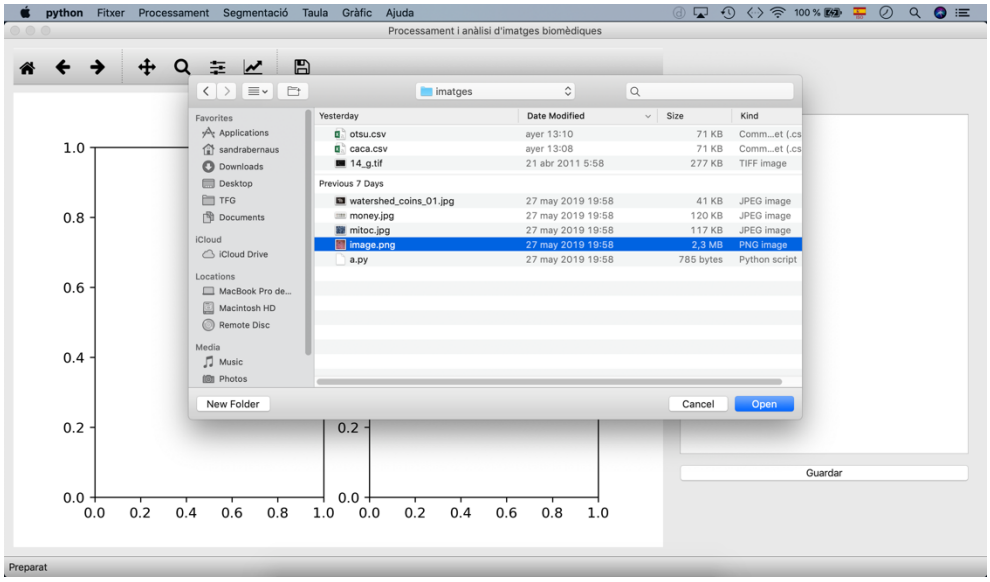


Fig 10: Finestra emergent per seleccionar fitxer.

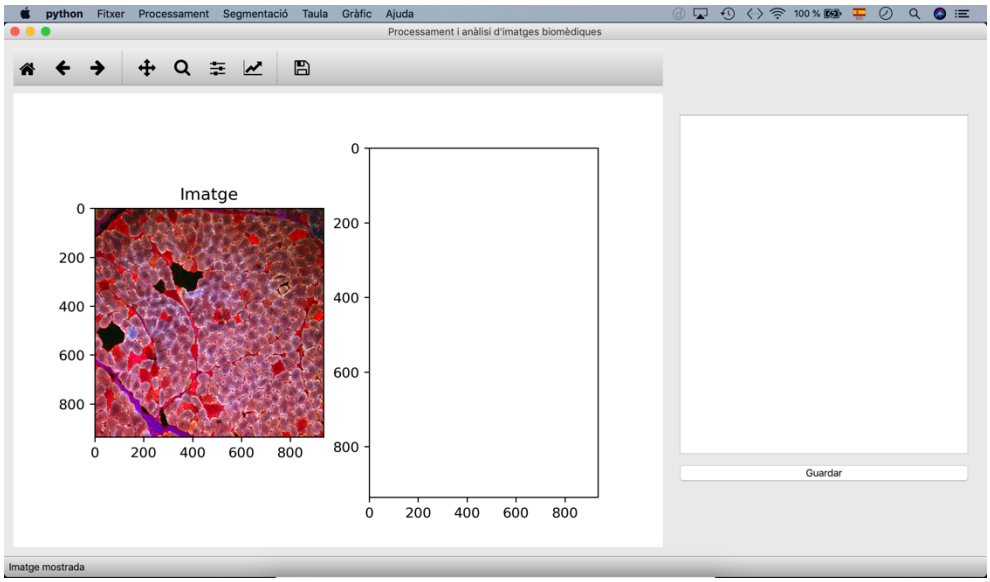


Fig 11: Disposició de la imatge seleccionada en pantalla.

En la figura 9, 10, 11 és mostra el flux que s'ha de seguir per a visualitzar una imatge. Seleccionant en l'opció del menú "Fitxer" amb el nom "Seleccionar imatge", s'obra una finestra emergent que permet a l'usuari seleccionar la imatge i mostrar-la en la finestra. Tot i que la imatge que s'ha mostrat és de format png, també es comprova que les imatges jpg i tif es puguin visualitzar en les figures 12, 13, 14 i 15.

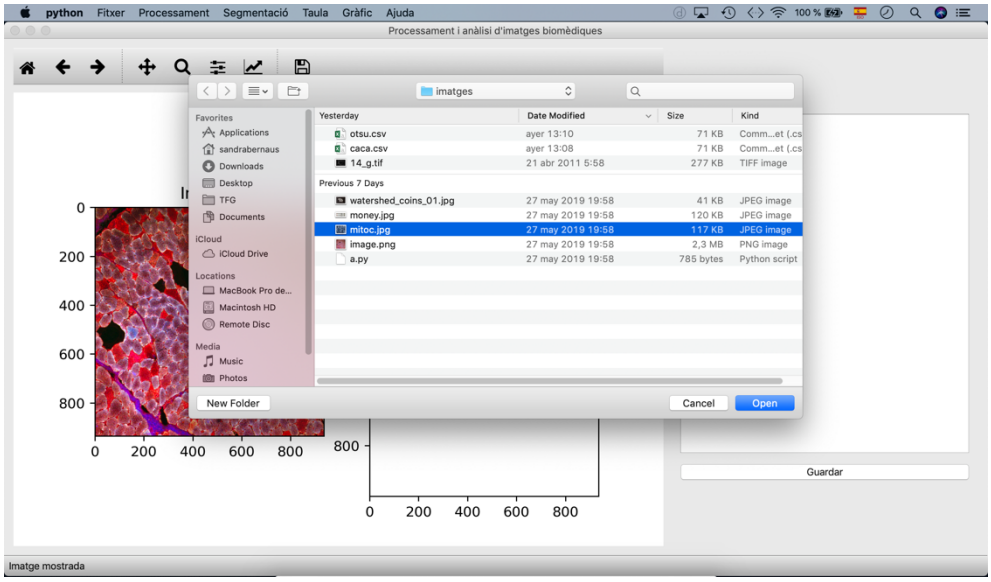


Fig 12: Selecció d'imatge png.

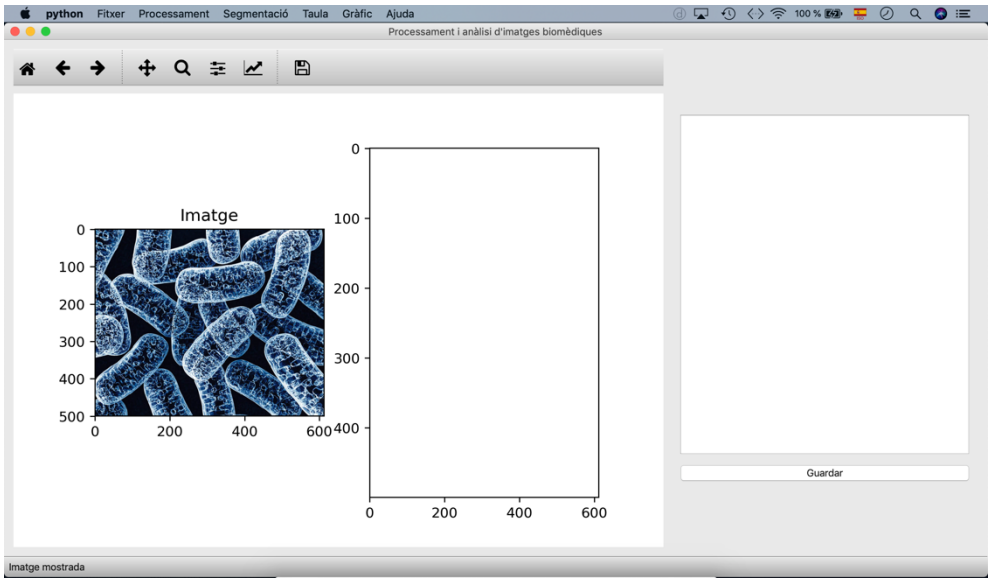


Fig 13: Visualització d'imatge png.

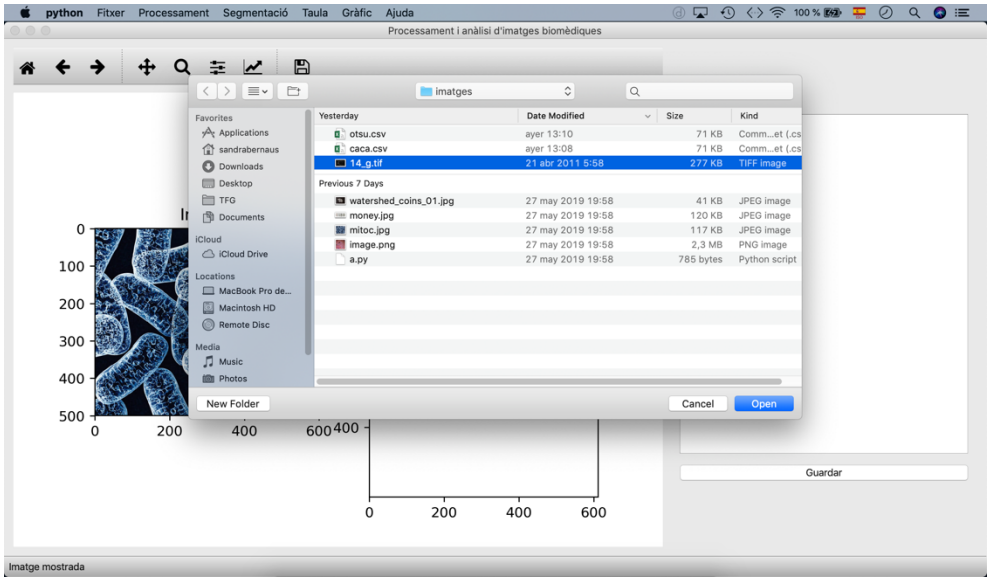


Fig 14: Selecció d'imatge tiff.

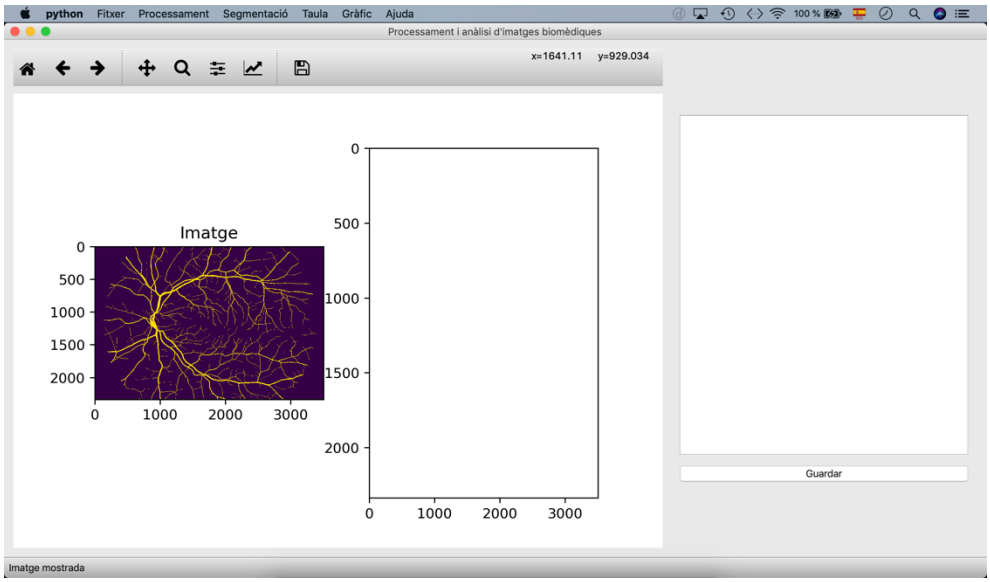


Fig 15: Visualització d'imatge tiff.

9.3 Test 3: Processament d'imatges

ID del test	T.03
Nom del test	Processament d'imatges.
Objectiu/Descripció del test	Demostrar que es poden aplicar diferents processats.
Criteri d'acceptació	El sistema permet processar la imatge seleccionada amb les opcions de menú de Processat.

Taula 12: Descripció del test 3.

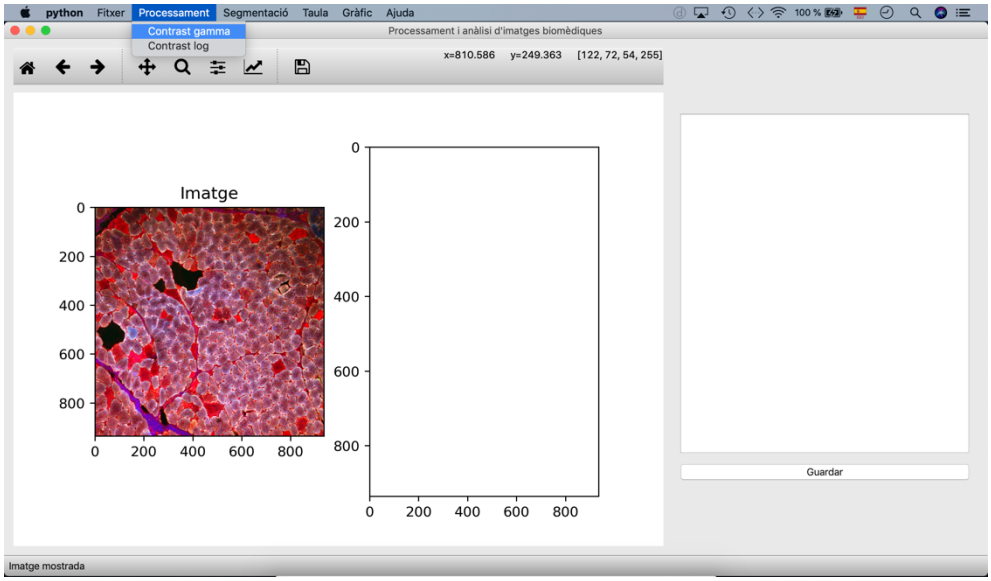


Fig 16: Acció per aplicar processat gamma.

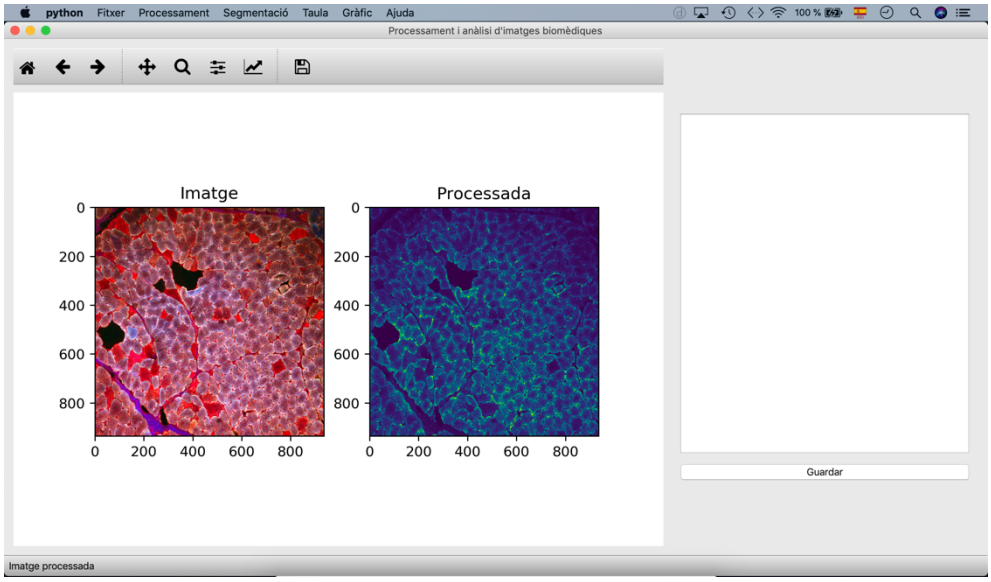


Fig 17: Aplicació del processat gamma i visualització.

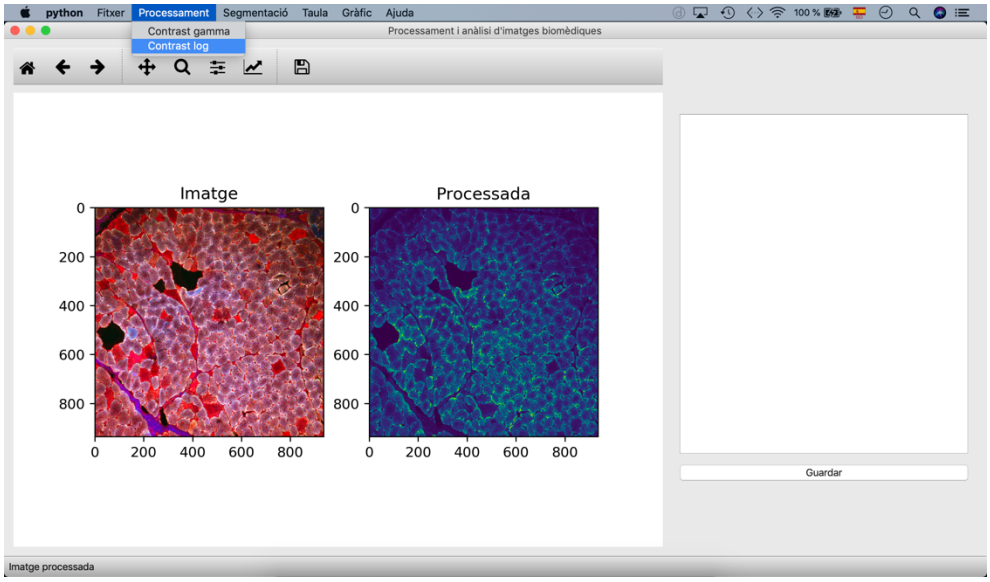


Fig 18: Acció per aplicar processat logarítmic.

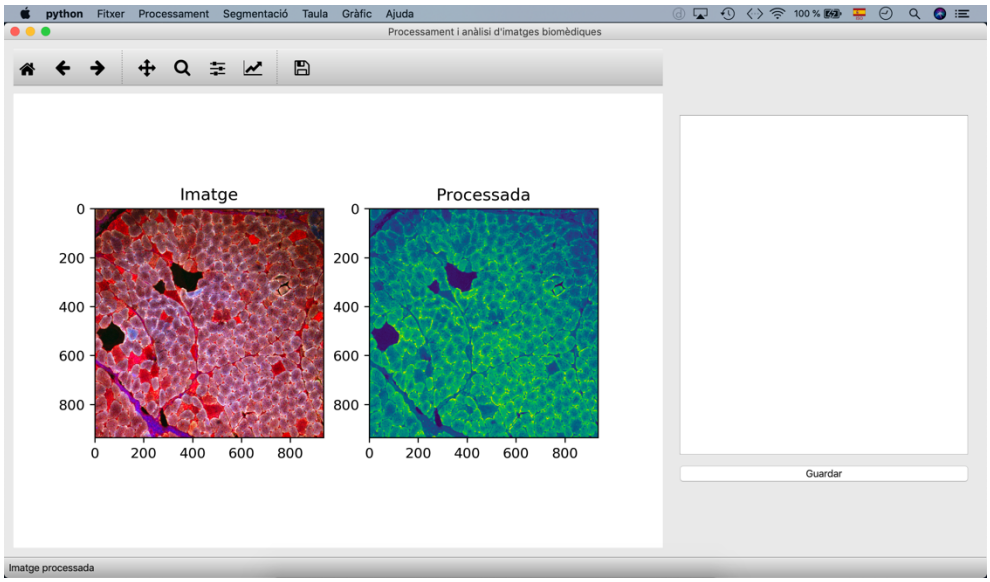


Fig 19: Aplicació del processat logarítmic i visualització.

En les figures mostrades en aquest apartat és es demostra la visualització dels diferents processaments mitjançant les opcions del menú “Processat”, en el segon eix del visualitzador d’imatge a l’esquerra de la finestra. En primer lloc el processament de correcció de contrast gamma en la figura 17 i en segon lloc el processament de correcció de contrast logarítmic en la figura 19.

9.4 Test 4: Segmentació d'imatges

ID del test	T.04
Nom del test	Segmentació d'imatges.
Objectiu/Descripció del test	Demostrar que es poden segmentar les imatges mitjançant diferents mètodes.
Criteri d'acceptació	El sistema permet segmentar la imatge seleccionada amb les opcions de menú de Segmentació.

Taula 13: Descripció del test 4.

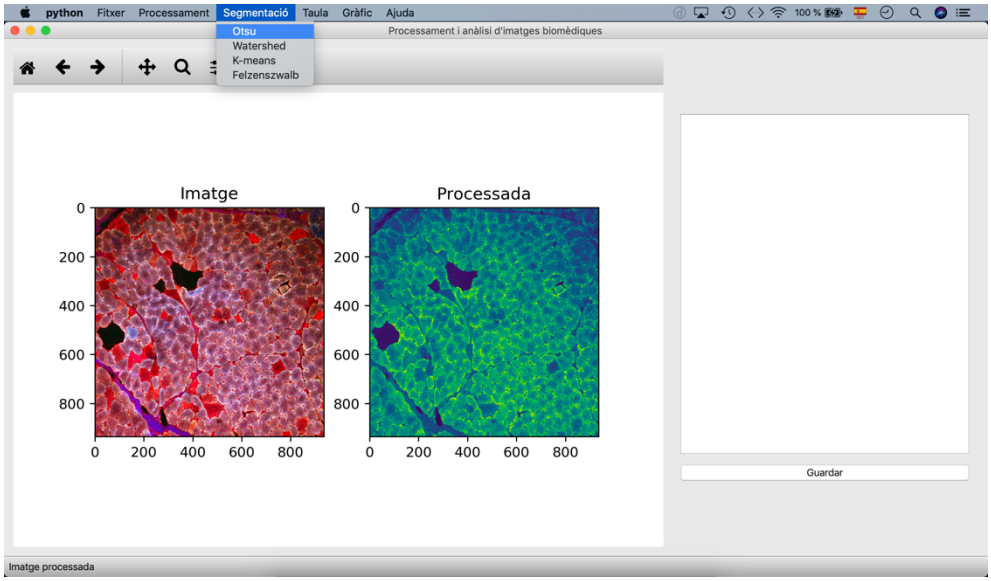


Fig 20: Acció de selecció de segmentació Otsu.

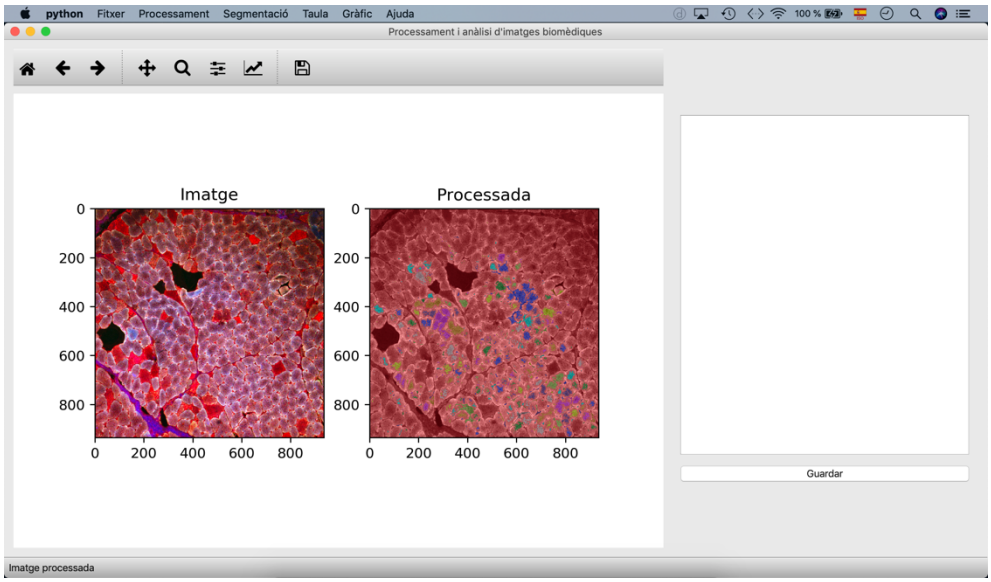


Fig 21: Aplicació de segmentació Otsu i visualització.

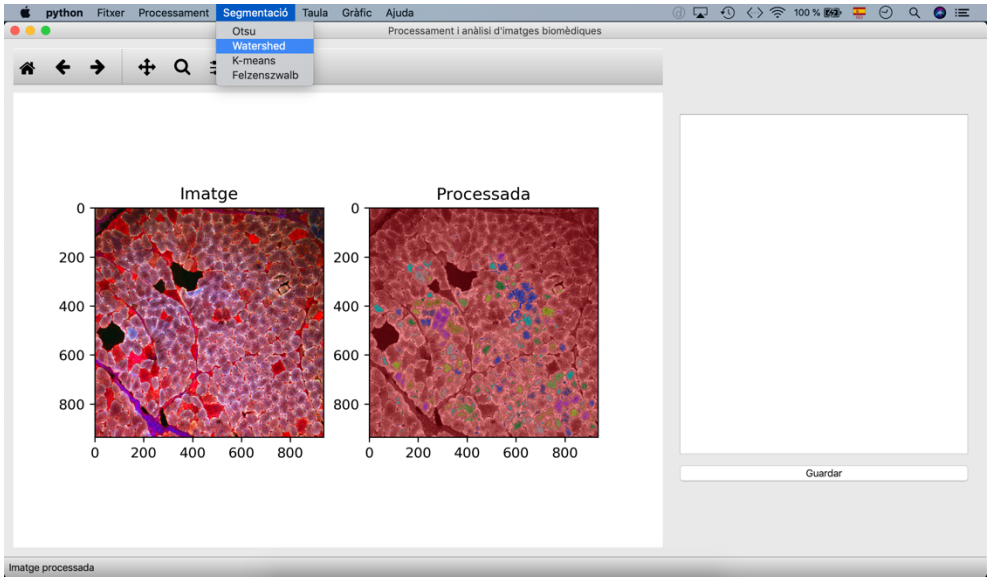


Fig 22: Acció de selecció de segmentació Watershed.

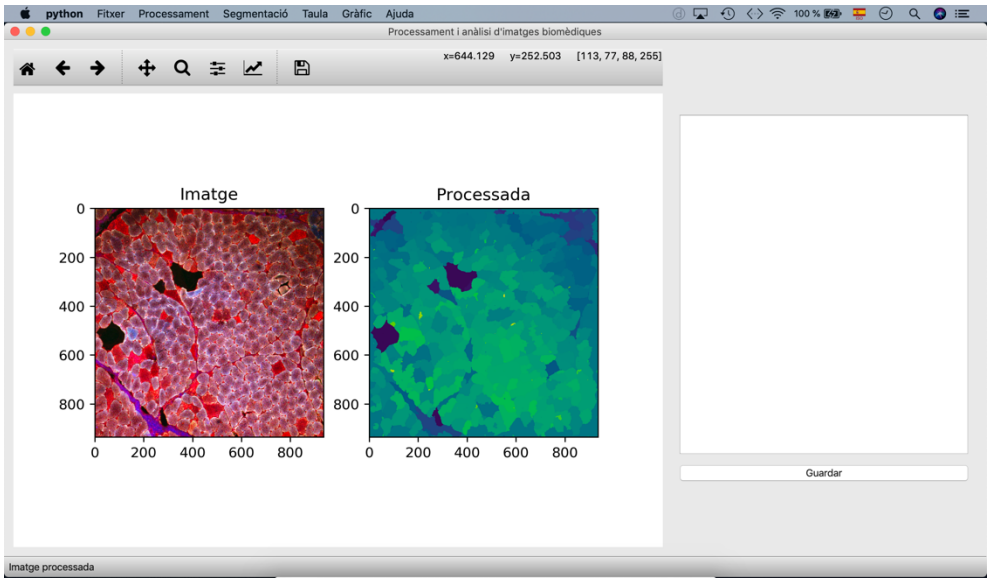


Fig 23: Aplicació de segmentació Watershed i visualització.

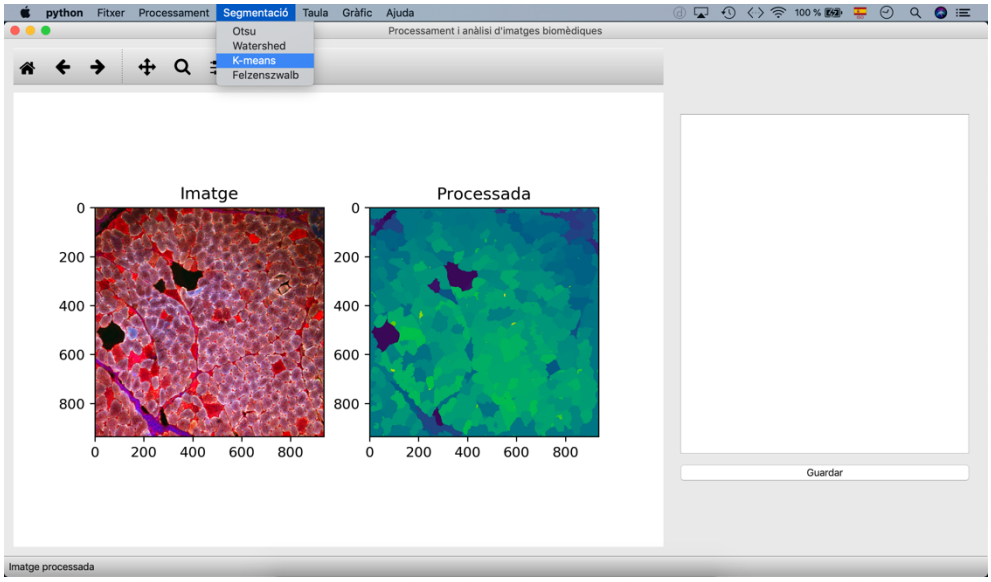


Fig 24: Acció de selecció de segmentació K-means.

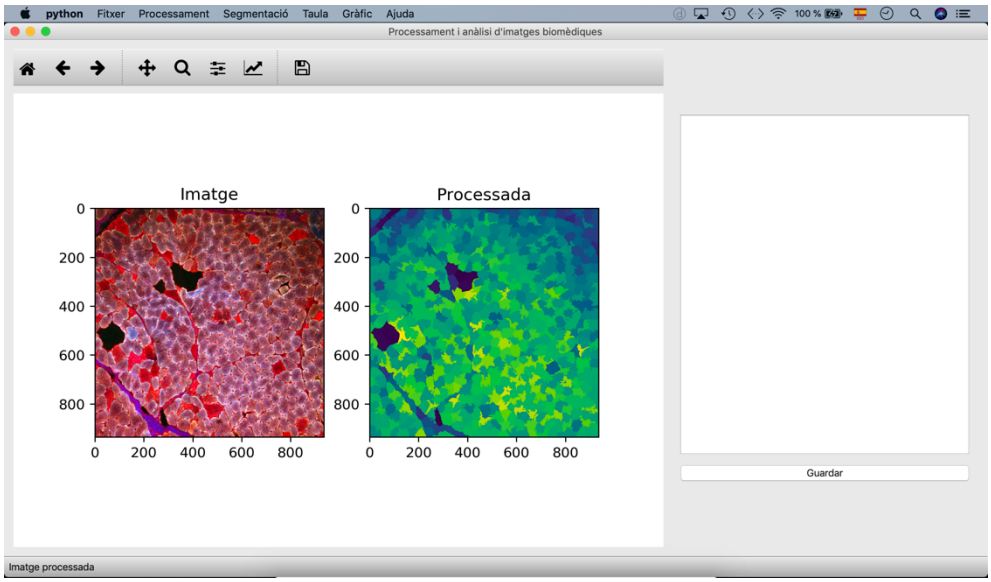


Fig 25: Aplicació de segmentació K-means i visualització.

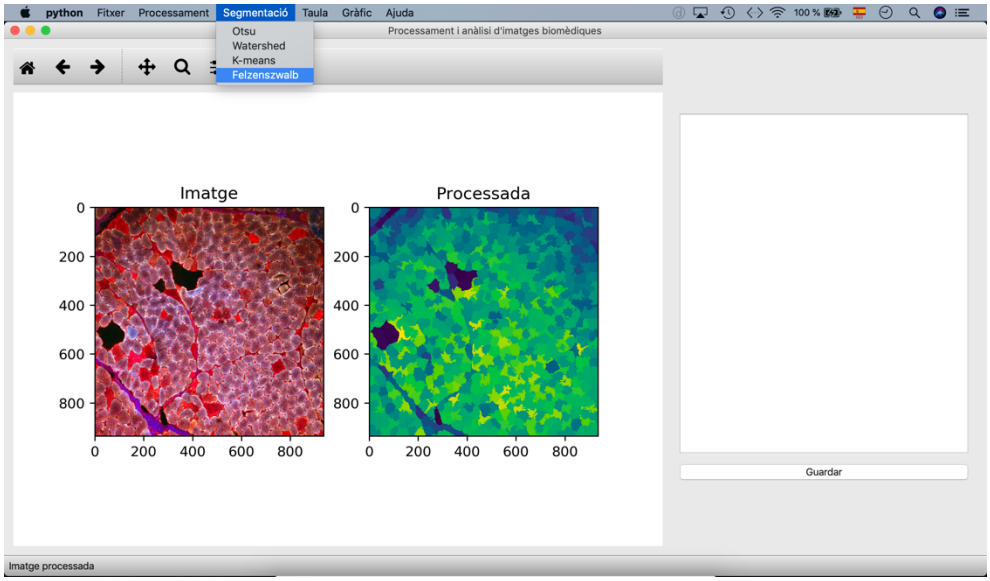


Fig 26: Acció de selecció de segmentació Felzenszwalb.

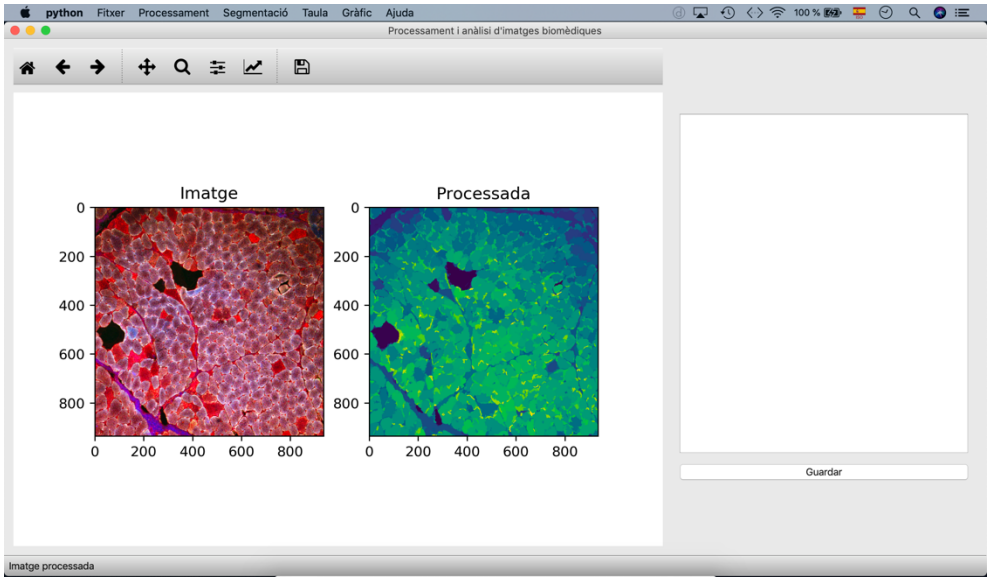


Fig 27: Aplicació dee segmentació Felzenszwalb i visualització.

En el test 6 s'ha demostrat com mitjançant les opcions de menú "Segmentació" s'apliquen les transformacions pertinents en la imatge processada de la banda dreta. L'aplicació conté 4 tipus de segmentacions no acumulables, de tal forma que al aplicar-n'hi una és mostra la segmentació aplicada, i si es decideix d'aplicar una altra segmentació no s'aplicarà per sobre de la imatge segmentada, sinó sobre la original. La figura 21 és mostra el resultat de la segmentació Otsu, en la figura 23 la segmentació per Watershed, en la figura 25 la segmentació K-means i finalment en la figura 27 la segmentació Fezenswalb.

9.5 Test 5: Eixos lligats i dades en cursor

ID del test	T.05
Nom del test	Eixos lligats i dades de cursor.
Objectiu/Descripció del test	Demostrar que els eixos de les dos imatges estan lligats, es a dir, que qualsevol modificació feta a una imatge, afectarà de la mateixa manera a l'altra. També es pot visualitzar en la barra d'eines la posició i el valor del píxel on reposa el cursor i fent un clic en la imatge les dades de la posició del cursor es visualitzaran en una bombolla.
Criteri d'acceptació	Les dos imatges tindran els eixos lligats i la possibilitat de mostrar la posició del cursor en una bombolla de diàleg i en la barra d'eines.

Taula 14: Descripció del test 5.

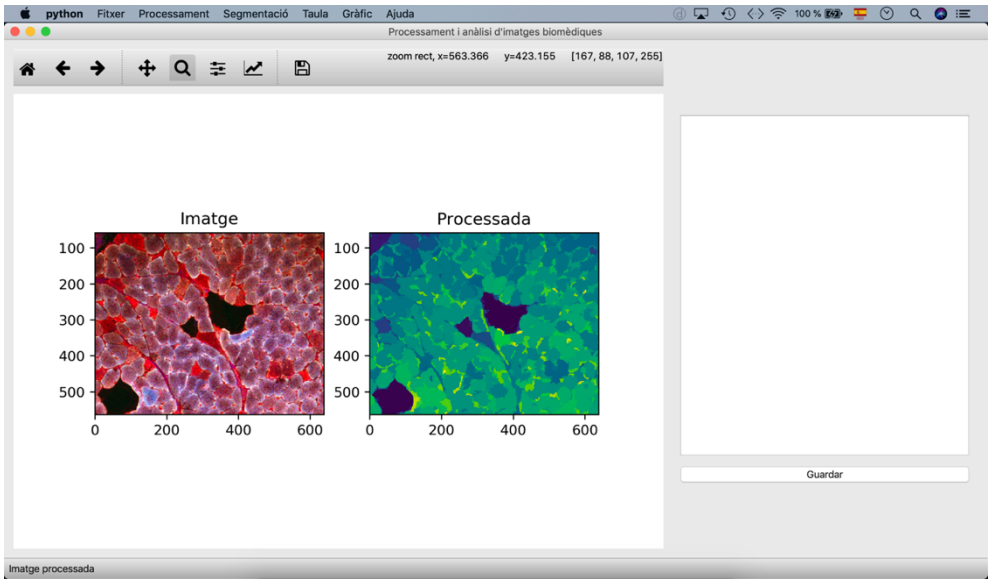


Fig 28: Zoom de la imatge origina, obtenint el mateix augment en la imatge processada.

En aquesta pantalla es pot veure com s'han ampliat les dos imatges paral·lelament, en seleccionat la lupa, que és l'eina per ampliar i reduir les dimensions de les imatges, s'ha creat un rectangle amb la secció de la imatge que es desitja veure de més a prop. De tal forma que es disposa la secció ampliada en la finestra de totes dos imatges.

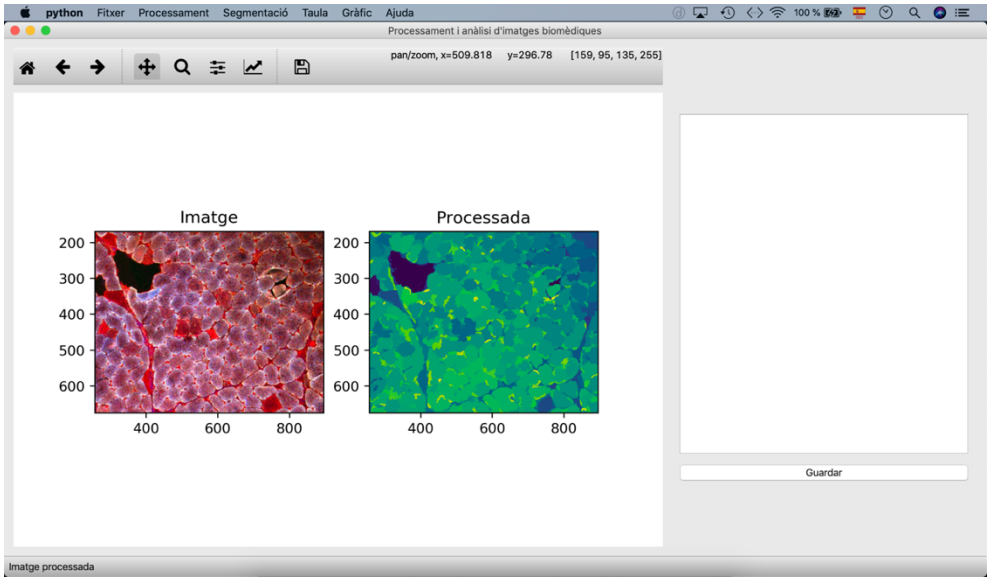


Fig 29: Moviment dels eixos lligat de les imatges.

En la figura 29 es demostra com, a través de la creu amb puntes de fletxa, podem desplaçar la imatge per a poder visualitzar la zona de la imatge que es desitgi, de tal forma que, tot i estar movent una imatge l'altra es mou igual.

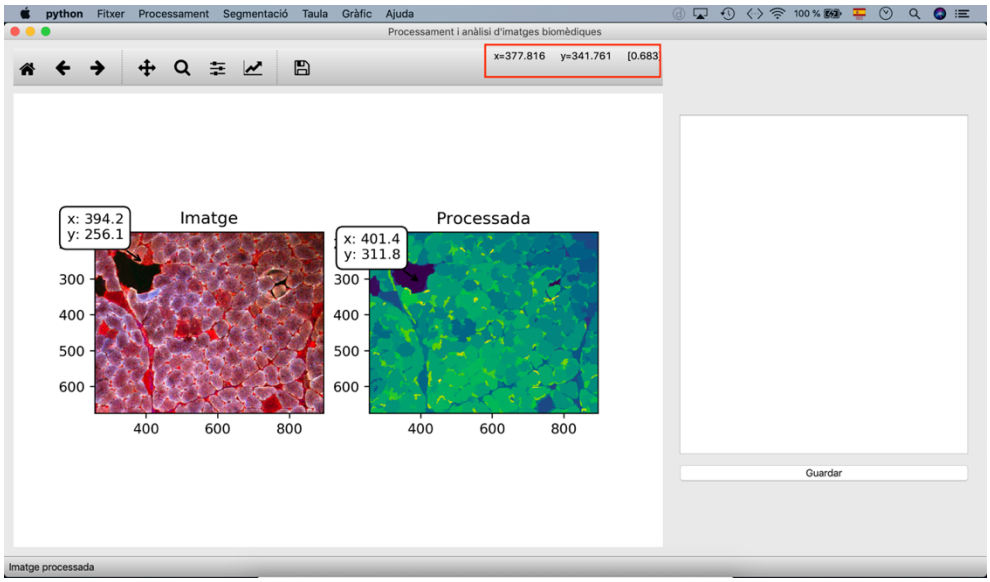


Fig 30: Visualització de les coordenades en bombolla de diàleg i en la barra d'eines.

En la 30a figura s'observa quines són les dues opcions per visualitzar les coordenades del ratolí en la pantalla. En la part dreta de la barra d'eines es poden veure les coordenades i el valor del píxel on recau el punter. El segon s'obté interaccionant amb la imatge mitjançant un clic sobre el píxel que es vol identificar, donant els resultats en una bombolla de diàleg.

9.6 Test 6: Desament d'imatges

ID del test	T.06
Nom del test	Guardar imatges.
Objectiu/Descripció del test	Les imatges que es mostren en el widget central de matplot, es poden guardar amb l'opció de la barra d'eines, guardar.
Criteri d'acceptació	Es poden guardar les imatges que s'estan visualitzant.

Taula 15: Descripció del test 6.

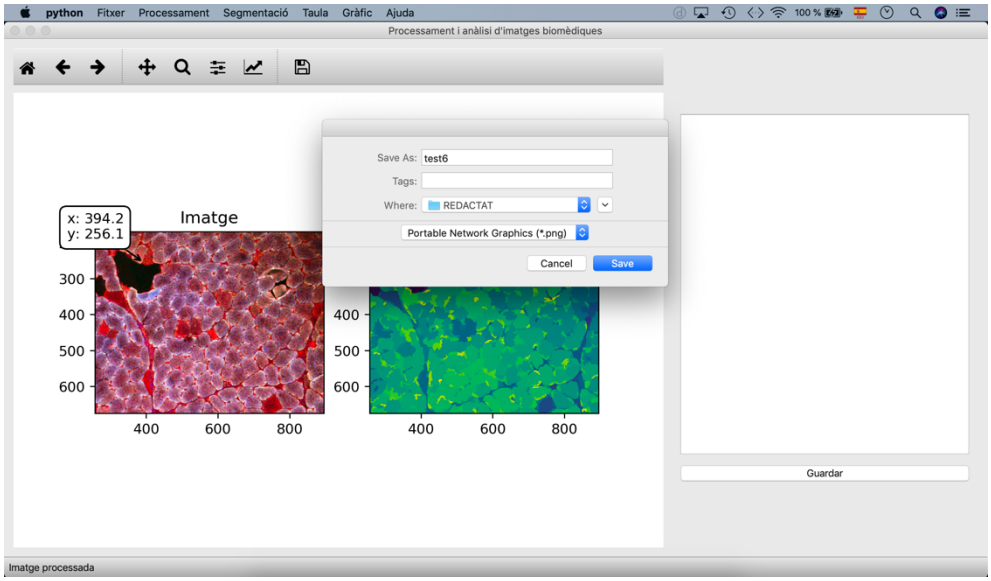


Fig 31: Guardar imatge amb l'opció de la barra d'eines de matplotlib.

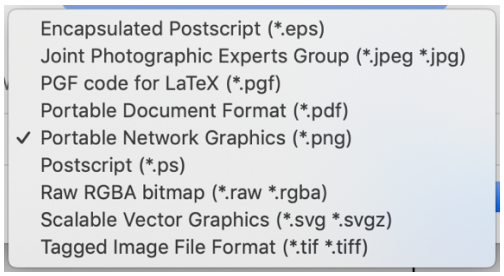


Fig 32: Possibles formats per guardar la imatge.

Per a guardar la imatge que es mostra en pantalla s'utilitza la icona del disquet, apareixent la finestra emergent que es veu en la figura 31, en la qual pots guardar el nom de la imatge, elegir l'etiqueta que se li vol assignar, el directori on es vol guardar i per últim el format (figura 32) en el qual es pot exportar.

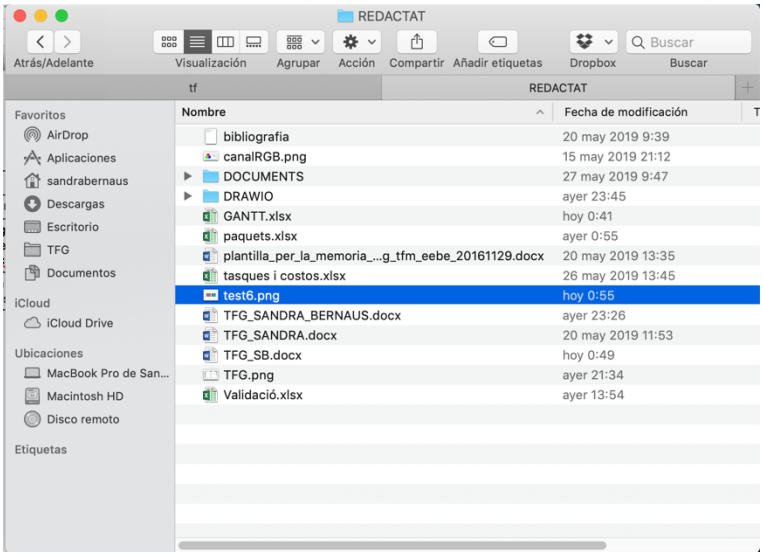


Fig 33: Fitxer guardat en el directori indicat.

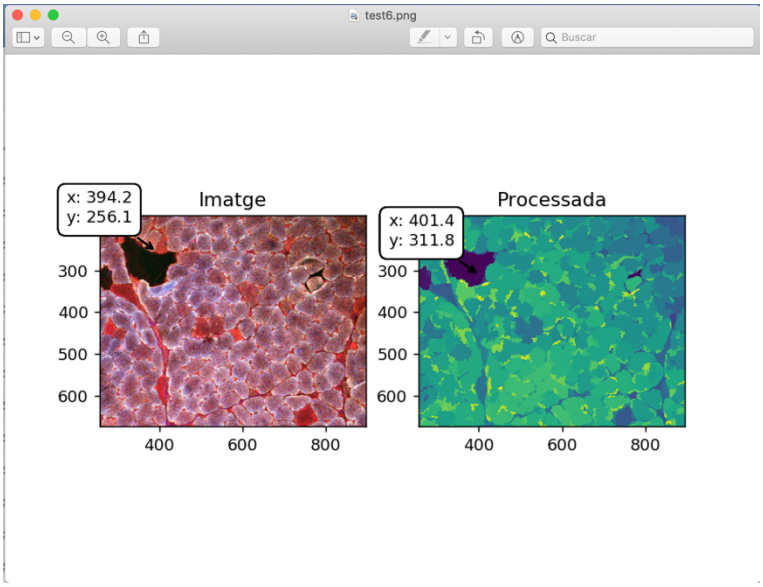


Fig 34: Imatge guardada correctament.

En la figura 33 i 34 es mostra com s'ha guardat la imatge en el directori corresponent, amb els eixos i les dades del cursor.

9.7 Test 7: Visualització i desament de característiques de cada regió

ID del test	T.07
Nom del test	Taula de propietats.
Objectiu/Descripció del test	Demostrar la visualització de la taula amb les propietats de cada segment. Es guarda les propietats en un fitxer csv.
Criteri d'acceptació	El sistema permet visualitzar la taula de propietats dels segments que s'ha dissenyat de totes les segmentacions. Es desa un fitxer csv amb la taula mostrada.

Taula 16: Descripció del test 7.

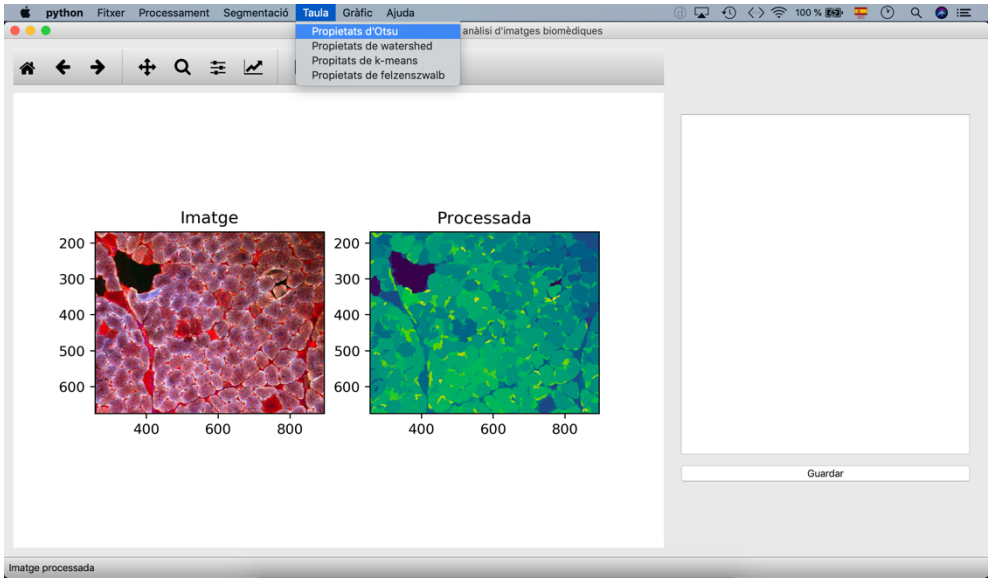


Fig 35: Acció per mostrar les propietats de les regions de la segmentació Otsu en la taula.

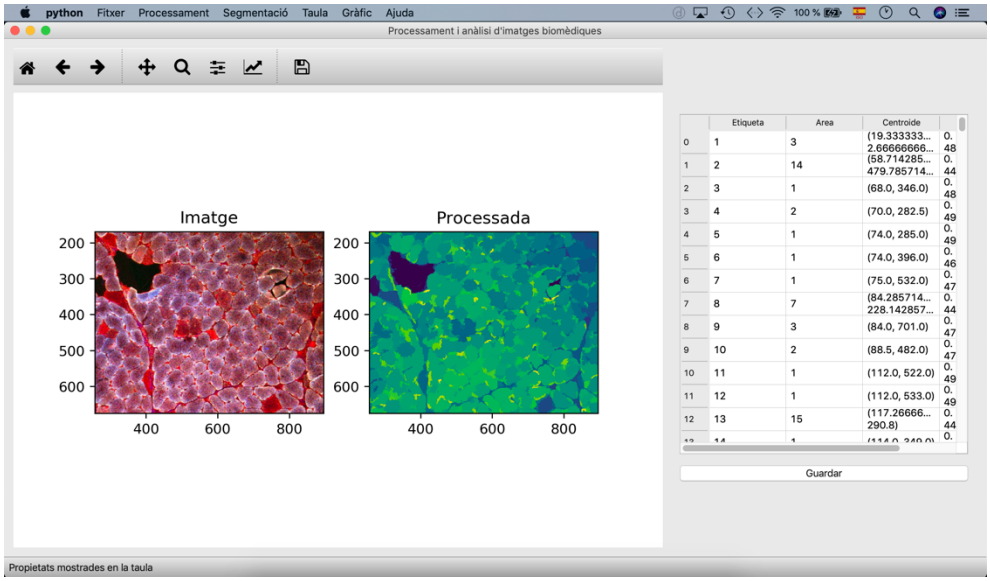


Fig 36: Visualització de les propietats les regions segmentades per Otsu en la taula.

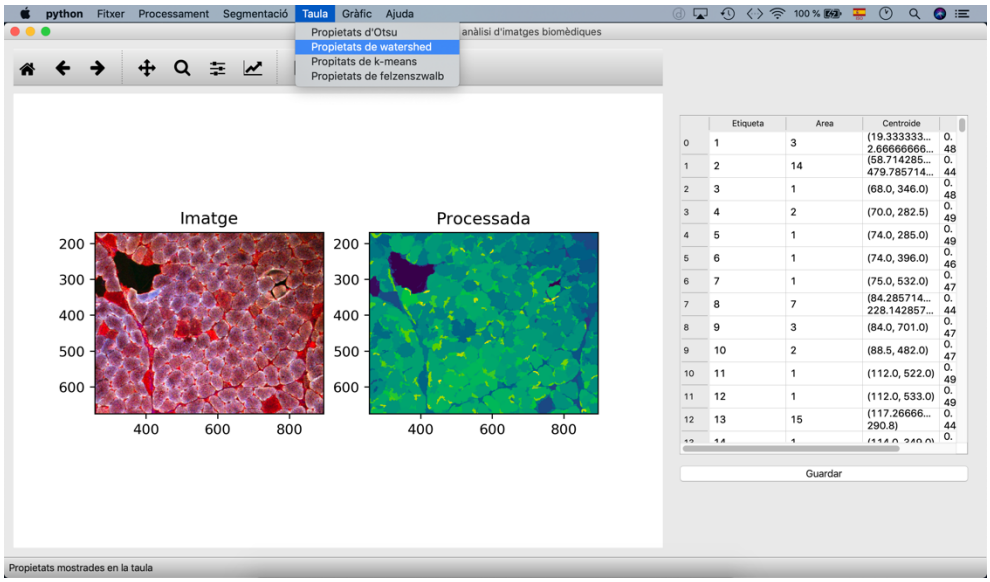


Fig 37: Acció per mostrar les propietats de les regions de la segmentació Watershed en la taula.

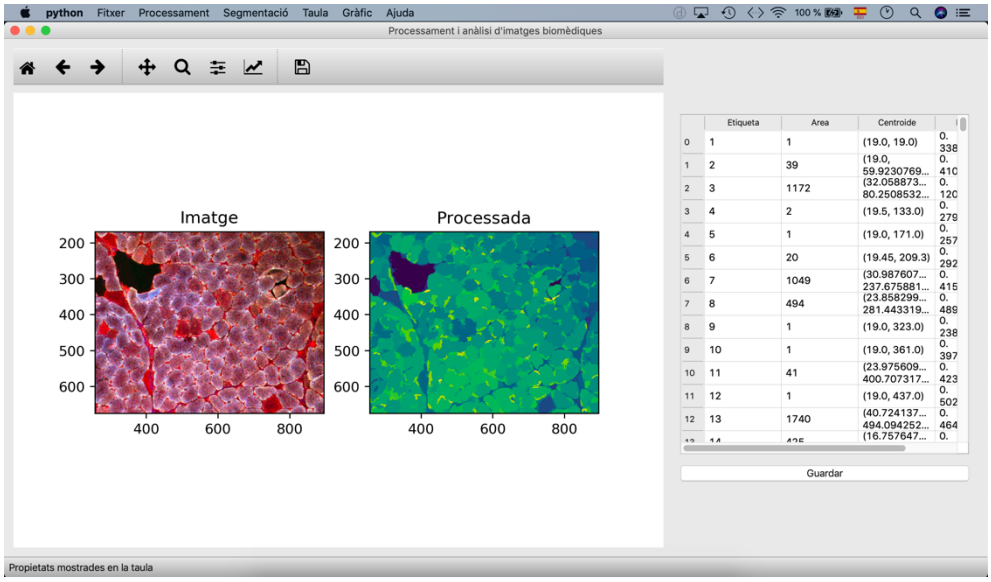


Fig 38: Visualització de les propietats les regions segmentades per Watershed en la taula.

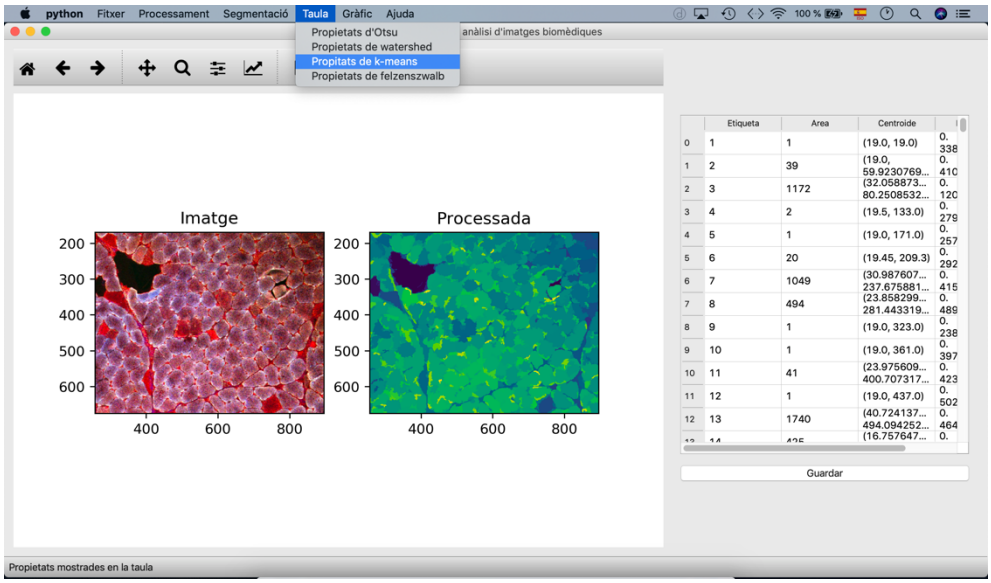


Fig 39: Acció per mostrar les propietats de les regions de la segmentació K-means en la taula.

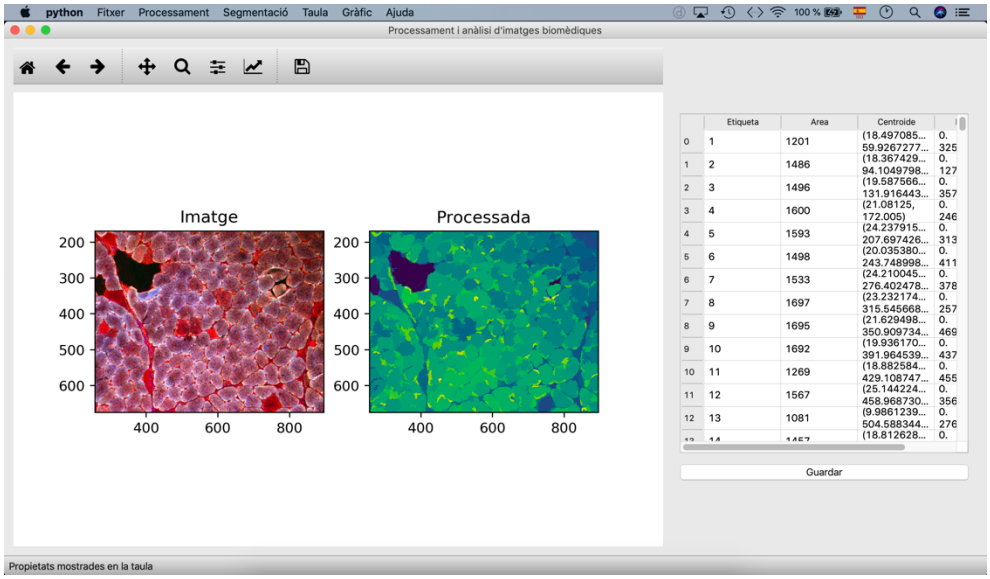


Fig 40: Visualització de les propietats les regions segmentades per K-means en la taula.

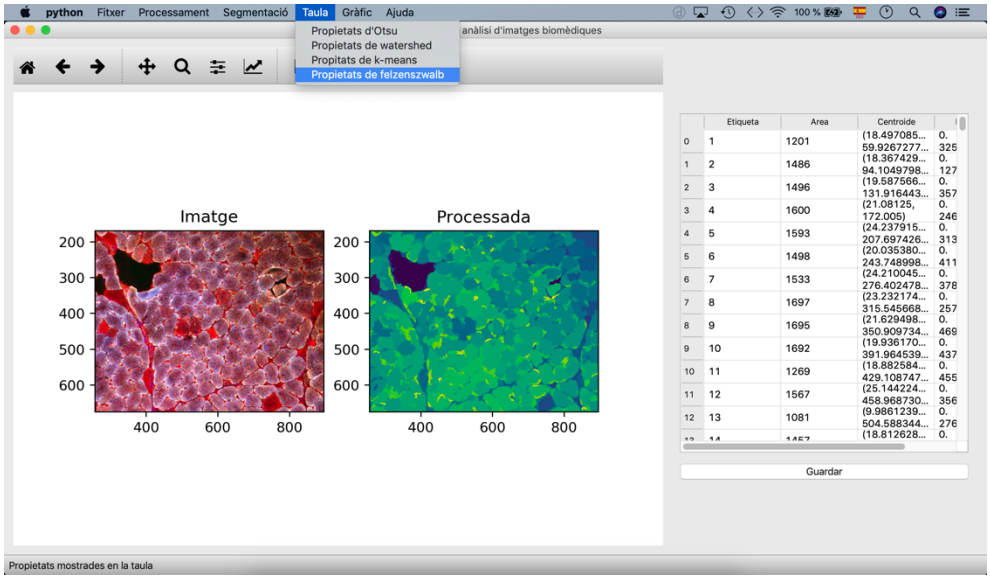


Fig 41: Acció per mostrar les propietats de les regions de la segmentació Felzenszwalb en la taula.

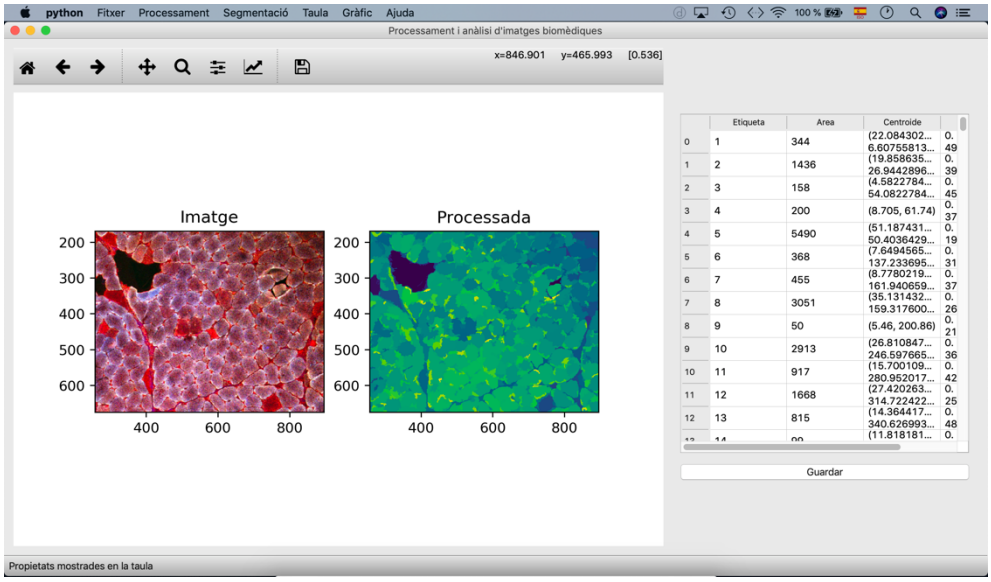


Fig 42: Visualització de les propietats les regions segmentades per Felzenszwalb en la taula.

En les figures del test 7 que s'han pogut observar, és mostra el flux que s'ha de seguir per visualitzar les característiques de cada regió que ens proporcionen els diferents processaments de segmentació. En primer lloc es selecciona el menú "Taula" i un cop dins es tria la segmentació aplicada a la imatge per a visualitzar les regions que ha definit i les propietats que les acompanyen. No és necessari les propietats extretes siguin de la imatge processada en pantalla.

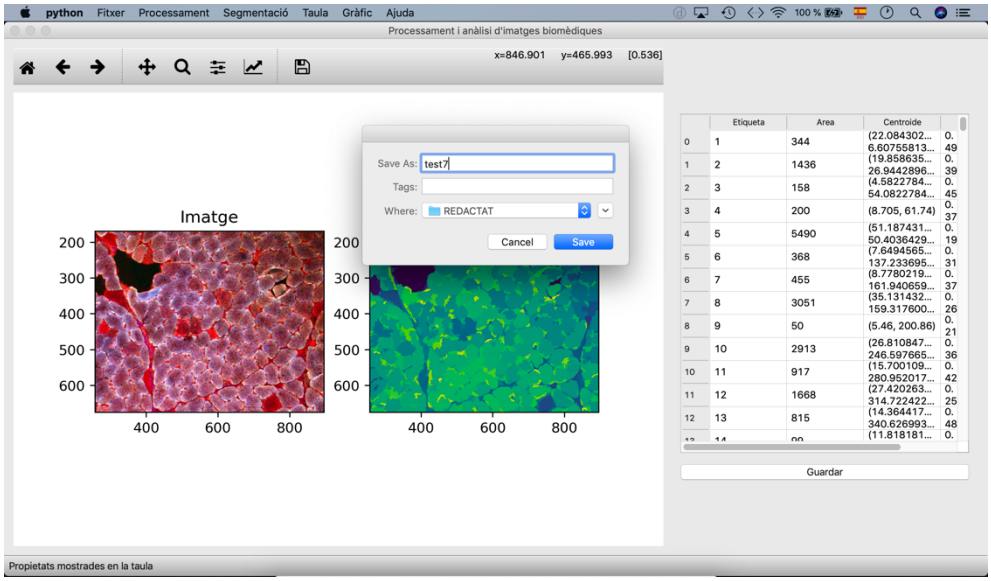


Fig 43: Guardar les dades de la taula en format csv.

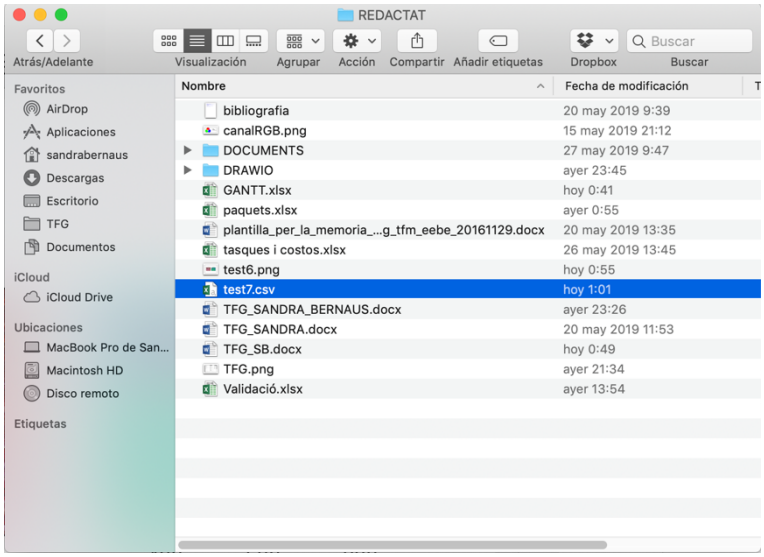


Fig 44: CSV guardat en el directori indicat.

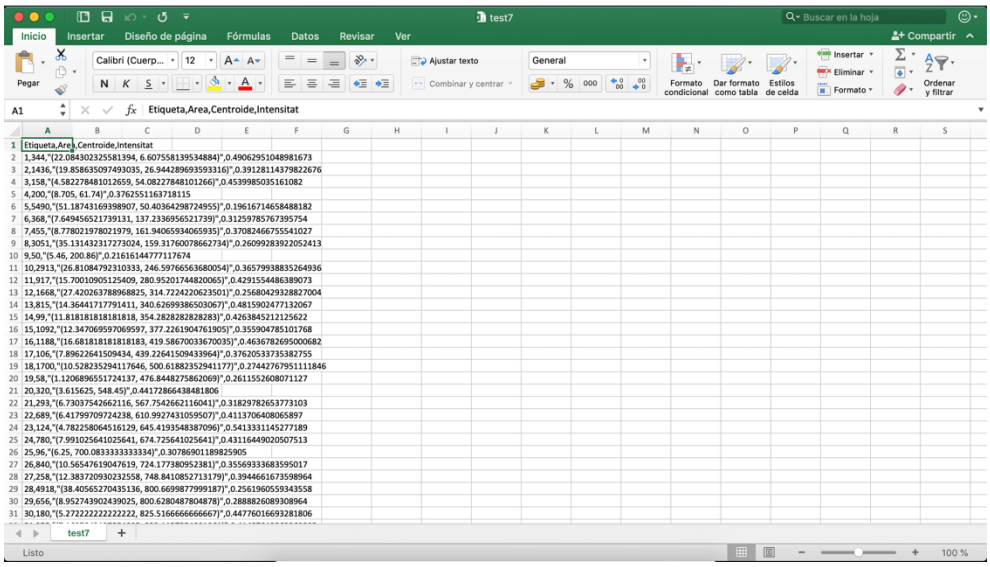


Fig 45: Dades guardades en el csv.

La figura 43, 44, 45 és el flux que es segueix per a emmagatzemar la informació de les taules en csv. En la primera figura és mosta la finestra emergent que apareix un cop es fa clic sobre el botó “Guardar”. Les opcions que ens deixa triar aquesta finestra són les mateixes que les imatges, excepte el format csv per defecte en que es guardaran les dades. En la segona imatge es mostra com es guarda correctament, en el directori i el nom que s’ha indicat. En la tercera s’observa les dades guardades en format csv i visualitzades en un Excel.

9.8 Test 8: Gràfic de propietats de cada regió

ID del test	T.08
Nom del test	Gràfics de propietats.
Objectiu/Descripció del test	Demostrar la visualització del gràfic en una finestra emergent que s'ha dissenyat per veure l'àrea vs la intensitat de cada segment.
Criteri d'acceptació	S'obra una finestra emergent amb un gràfic scatter de l'àrea entre la intensitat de la segmentació per llindar Otsu.

Taula 17: Descripció del test 8.

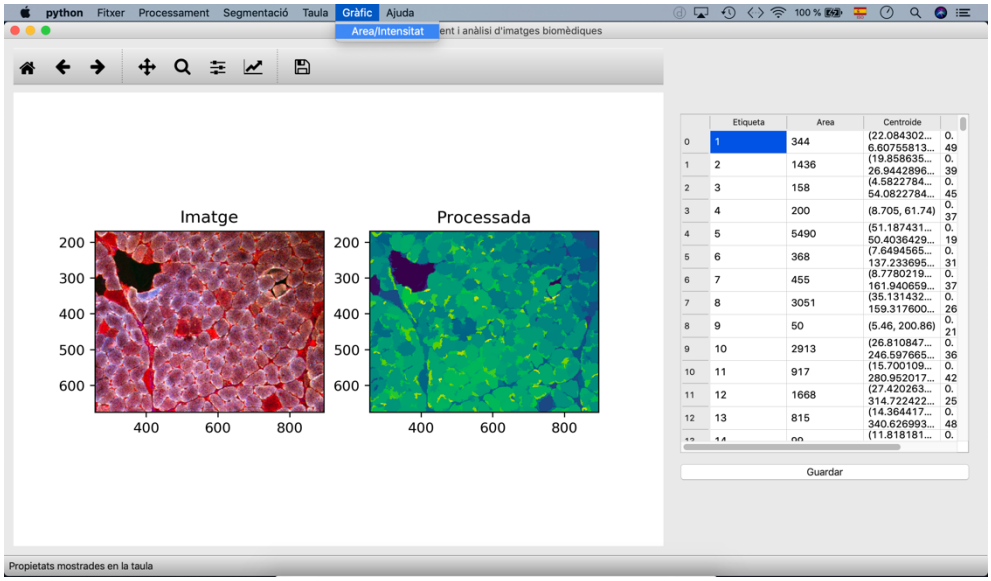


Fig 46: Acció per a la visualització de l'àrea vs intensitat de les regions extretes per la segmentació Otsu.

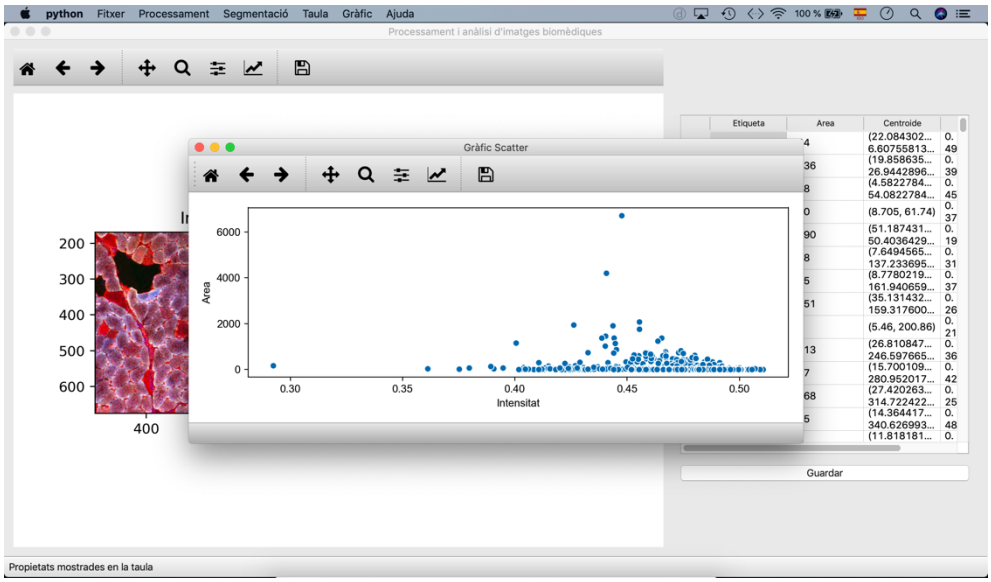


Fig 47: Visualització de l'àrea vs intensitat de les regions extretes per la segmentació Otsu.

En aquest test es pot veure com s'obra una finestra emergent amb un gràfic de les propietats de cada segment extret per segmentació Otsu, on es distribueixen els punts segons l'àrea i la intensitat. La finestra emergent és de matplotlib, per tant permet, igual que en les imatges de la finestra central, ser guardada.

9.9 Test 9: Sobre el programa

ID del test	T.09
Nom del test	Sobre el programa.
Objectiu/Descripció del test	Demostrar que existeix una opció de menú anomenada 'Sobre el programa' que obra una finestra emergent amb la informació del programa.
Criteri d'acceptació	S'obra una finestra emergent amb la informació del programa.

Taula 18: Descripció del test 9.

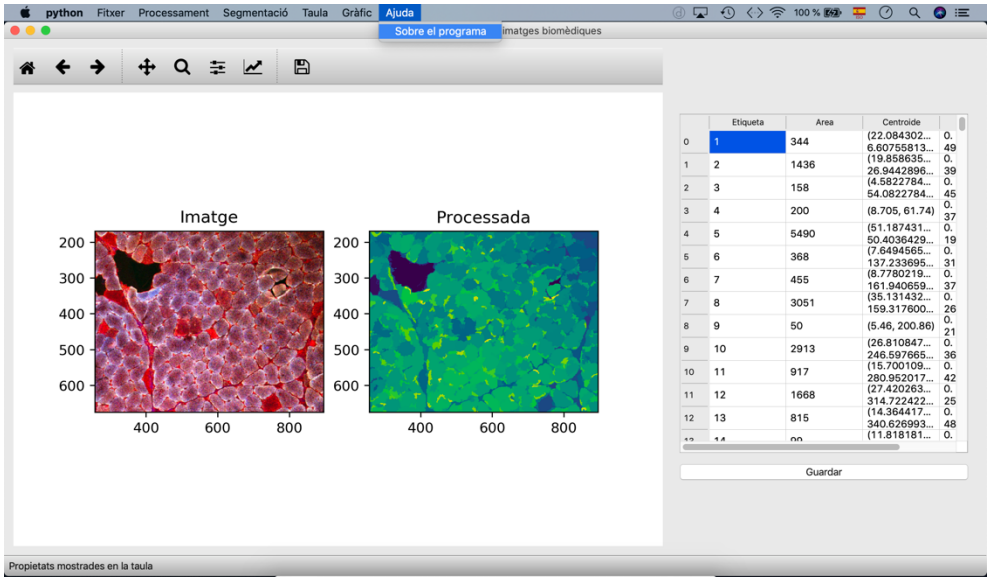


Fig 48: Acció per a visualitzar la informació sobre el programa.

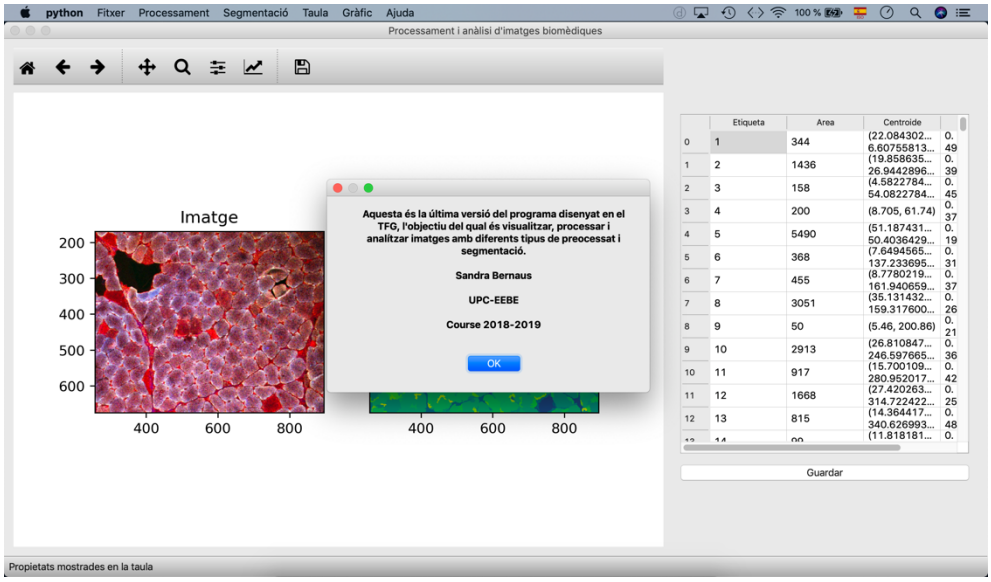


Fig 49: Visualització de la informació del programa en una finestra emergent.

En la figura 48 i 49 es pot comprovar com es visualitza la informació del programa al seleccionar l'opció "Sobre el programa" del menú "Ajuda".

9.10 Test 10: Tancar el programa

ID del test	T.10
Nom del test	Sortir del programa.
Objectiu/Descripció del test	Demostrar l'existència d'una opció de menú que ens permet sortir del programa. En fer-ho surt una finestra emergent de confirmació i en acceptar es surt del programa.
Criteri d'acceptació	S'obra una finestra emergent en fer clic en "Sortir del programa". En confirmar el programa es tanca.

Taula 19: Descripció del test 10.

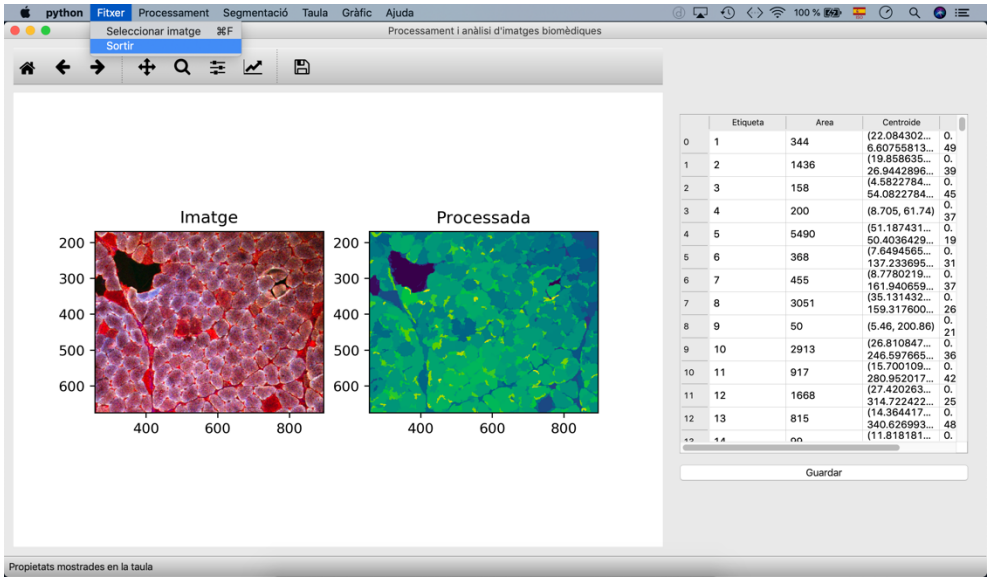


Fig 50: Acció per sortir del programa.

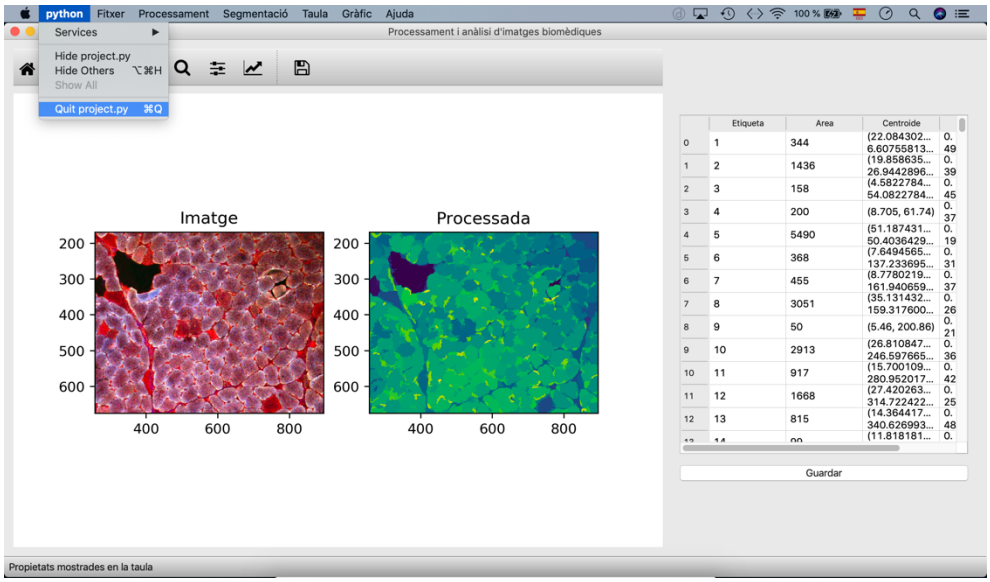


Fig 51: Segona opció per sortir del programa i indicació de la abreviació 'cmd+Q' o en cas de Windows 'ctr+Q'.

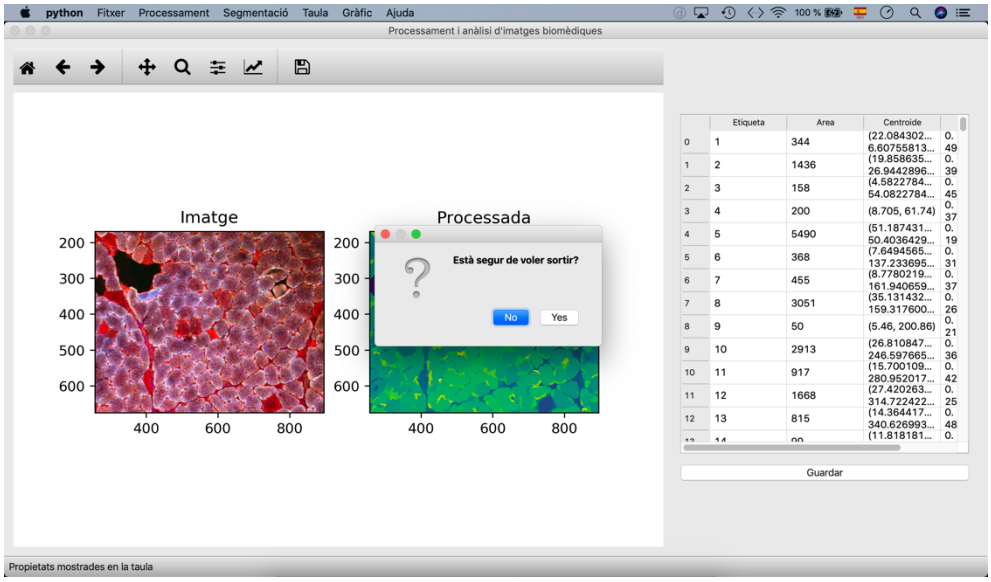


Fig 52: Finestra emergent demanant confirmació per a tancar el programa.

En aquest últim test es comprova la sortir de l'aplicació i les possibles opcions per fer-ho. En la figura 50 es mostra com sortir mitjançant l'opció "Sortir" del menú "Fitxer". En la figura 51 es visualitza com sortir del programa Python per defecte, amb la drecera de teclat "cmd+q" per a MacOs i "ctrl+q" per a Windows i Linux. I en la figura 52 es mostra la finestra emergent de confirmació en intentar sortir del programa, la qual serveix per assegurar que no ha estat una equivocació.

10 Continuació del Treball

Aquest programa ha estat l'inici del disseny d'una plataforma per utilitzar en els centres de recerca mèdica i per tant, es pretén implementar millores, ja que aquest apartat serà l'etapa de manteniment que tot programa de software necessita per a seguir en ús. En aquest treball s'ha dissenyat la base del programa, però es volen implementar noves funcions com:

1. Permetre a l'usuari seleccionar una regió de la imatge i mostrar a quina etiqueta pertany la selecció i les propietats extretes d'aquesta.
2. Compatibilitzar el programa amb imatge DICOM utilitzades en els hospitals.
3. Inserir diferents codis de segmentació específics per a cada aplicació, un dels propòsits seria implementar el codi per observar el dèficit de complexos de la cadena respiratòria de la mitocòndria, que utilitza xarxes de convolucions neuronals, i que ha estat dissenyat per un company anomenat Mauricio Shoji.
4. Poder carregar més d'una imatge a la vegada i fer la selecció dels mateixos filtres a les imatges d'una mateixa mostra. En aquest cas també s'haurà de tenir en compte l'extracció de dades en taules diferents i gràfiques a part, cadascuna amb el nom de la imatge corresponent.

11 Gestió econòmica

11.1 Pressupost de recursos humans

Com ja s'ha comentat anteriorment hi haurà un únic treballador en l'empresa que durà a terme 5 tipus de rols diferents, aquest treballador serà una jove enginyera sense experiència i per tant cada hora realitzada cobrarà el mateix. Segons *El diari dels Enginyers Industrials de Catalunya* (COEIC/AEIC, 2017) el salari mitja d'un enginyer jove sense experiència seria d'uns 22.000 € a l'any per jornada completa de 8 h.

Rol	Mesos	Preu (€/mes)	Cost
Jove enginyera	5	1.833,33	9166,65
Total			9166,65

Taula 20: Pressupost de recursos humans

11.2 Pressupost de hardware

En el projecte s'ha utilitzat únicament un element de hardware, com ja s'ha comentat en l'apartat 3.3.2. El cost d'aquest producte es reflecteix en la taula 22. L'amortització es considerarà que és el preu del producte entre els anys de vida útil i les hores d'utilització per al projecte, considerades 720 h.

$$\boldsymbol{Amortització} = Preu \cdot (anys\ de\ vida\ útil \cdot Hores\ laborables\ a\ l'any)$$
$$= 1.755{,}59 \div (5 \cdot 2000) = \mathbf{0{,}175559}$$

Producte	Preu (€)	Vida útil (anys)	Amortització (€/h)	Hores d'ús	Cost
MackBook Pro 2014	1.755,59	5	0,18	720,00	126,40
Total					126,40

Taula 21: Pressupost del Hardware utilitzat en el projecte.

11.3 Pressupost de software

Els softwares utilitzats en aquest projecte, són programari lliure excepte Microsoft Office la llicència del qual s'ha obtingut per 69,00 €.

Producte	Preu (€)
Microsoft Office	69
Total	69

Taula 22: Pressupost del Software utilitzat en el projecte.

11.4 Costos indirectes

En els costos indirectes del projecte s'inclouran totes aquelles despeses relacionades amb el projecte de manera indirecta, com ara l'electricitat, la Internet, el lloc de treball, el transport per a les reunions amb el director del projecte i una línia de mòbil per estar en contacte amb el mateix mitjançant missatgeria instantània.

Per fer el càlcul del consum d'electricitat s'utilitza el consum energètic mig del Macbook Pro, el preu de l'electricitat en les hores punta amb la companyia Endesa i les hores d'utilització de l'ordinador que són respectivament 63,5 Wh (Electrocalculator, 2018), 0,1596 €/kWh (Selectra, 2018) i 684 h considerant que l'ordinador s'utilitza un 95% de les hores dedicades al projecte. D'aquesta forma en la següent equació es calcula el consum d'electricitat total consumit durant el projecte.

$$0{,}0635kWh * 0.1596\frac{€}{kWh} * 684h = 6{,}93€$$

La tarifa d'Internet aplicada per la companyia és d'uns 42 € al mes, utilitzada durant 5 mesos.

L'espai de treball considerarem que equivaldrà al temps d'utilització del pis on resideix l'autor del treball, i suposant que es cobraria el valor del preu del lloguer per hores, el lloguer costa 400 € al mes, que suposaria 0,56 €/h. Les hores utilitzades seran un 90% del total d'hores dedicades al projecte i això suposarà 360 € totals.

Tenint en compte que els transports s'han realitzat d'anada i de tornada fina a l'ETSEIB cada dilluns durant 4 mesos, ja que s'han fet fins al dia d'entrega del TFG, no s'inclouran els dies fins a la presentació. Els desplaçaments s'han realitzat amb els bitllet de transport públic de 10 viatge (T-10) i d'una zona, amb el preu de 10,20 €, tenint en compte que s'han realitzat 36 viatges la despesa haurà estat de 36,72 €.

Rol	Preu	Mesura d'utilització	Cost (€)
Electricitat	0,1596 €/kWh	63.5Wh durant 684h	6,93
Internet	42€/mes	5 mesos	210
Espai de treball	0,56€/h	648 hores	360
Transport	1,02€/viatge	36 viatges	36,72
Total			613,65

Taula 23: Cost de les despeses indirectes.

11.5 Costos per imprevistos

En qualsevol projecte s'ha de tenir en compte que hi poden haver imprevistos i desviacions, per tant s'ha de valorar un augment de costos sobre el salari dels treballadors, despeses en hardware i indirectes, el software s'ha calculat per llicència i per tant no afectarà en l'augment d'hores d'utilització. El principal risc del projecte és el retard en la finalització del programa, en qualsevol de les etapes, en el desenvolupament, gestió, presa de decisions, disseny, desenvolupament, correcció d'errors o finalització del projecte.

El pressupost del projecte ha estat calculat per tal d'assumir possibles desviacions del 15% en totes les tasques, indicant que les hores efectuades serien de 720 en comptes de les 612 que s'han considerat per fer el projecte.

11.6 Pressupost total

Fent el recompte total dels costos del projecte aquest ascendiria a 9976,7 €.

Despesa	Dedicació (h)
Recursos humans	9166,65
Recursos de hardware	126,4
Recursos de software	69
Costos indirectes	613,65
Costos imprevistos	0*
Total	9975,7

Taula 24: Pressupost total del projecte.

*Els costos imprevistos estan afegits a cadascun dels recursos i costos.

12 Anàlisi de l'impacte ambiental

Les accions humanes sempre tenen un efecte cap al medi ambient, en aquest projecte podríem considerar que no hi ha hagut influències significatives, ja que es tracta d'un desenvolupament d'un software on l'única eina de treball ha estat l'ordinador. No considerarem l'efecte de la fabricació de l'ordinador derivat del projecte, ja que aquest no s'ha comprat per a la seva realització. Però el consum d'energia sí que anirà relacionat amb aquest.

La producció d'energia sempre causa un efecte negatiu en el medi, sigui d'energia renovable o no. Tot i que l'efecte entre les dues no es pot arribar a comparar, la primera només influeix amb la fabricació de les eines/instal·lacions de captació de l'energia renovable i l'altra des de la fabricació de les instal·lacions, processos, emissions, etc. En el cas d'aquest projecte, s'ha utilitzat energia no renovable subministrada per Endesa.

Per altra banda, s'han realitzat 2 desplaçaments un dia a la setmana per fer les reunions periòdiques del seguiment del projecte, impartint un efecte negatiu al medi ambient amb les emissions de gasos contaminants despreses pel vehicle de transport autobús interurbà.

Conclusions

El resultat del projecte ha estat la creació d'una GUI senzilla i intuïtiva que servirà com a eina de suport en centres de recerca. S'ha aconseguit complir la majoria dels objectius plantejats a l'inici del projecte i els requisits establerts pels usuaris. L'aplicació és multiplataforma, permet la visualització, el processament i l'anàlisi de les imatges. A més a més, és una aplicació de codi obert que permet la implementació de noves funcionalitats en el programa, com ara nous processaments, segmentacions i tècniques d'anàlisi de dades. Per tant, l'aplicació permet extreure dades significatives de les regions trobades en les imatges, però encara no s'han pogut relacionar amb els biomarcadors que definiran la naturalesa del segment. Implícitament si s'utilitza l'aplicació es reduiria la transferència de dades entre investigador i expert en el processat d'imatges biomèdiques.

Per el moment l'aplicació consisteix en 2 pantalles, de les quals una és la principal i permet visualitzar la imatge original i la processada o segmentada en paral·lel, acompanyades d'una taula amb les propietats de cada regió. La següent pantalla és la destinada a l'anàlisi gràfic de les dades de les imatges.

El desenvolupament del software ha estat una tasca que ha necessitat una planificació i organització constant, ja que són moltes les fases que s'han de complir per a obtenir els resultats esperats. Gràcies a l'assignatura Projectes d'Enginyeria he pogut aplicar els meus coneixements sobre la gestió de temps, prioritats, etapes i fases per a obtenir un diagrama de Gantt que ha marcat el calendari d'entregues internes per a la finalització del treball a temps. En l'estudi del pressupost del projecte he pogut aplicar els coneixements apresos en dues assignatures Empresa i Projectes d'enginyeria.

S'ha necessitat una gran inversió de temps en l'aprenentatge de la programació orientada a objectes amb Python i de les llibreries que mai s'havien utilitzat com PyQt, scikit-image, matplotlib i seaborn. Però els coneixements de programació amb Python apresos en l'assignatura d'Informàtica han sigut de gran ajuda per a tenir unes bases sòlides on construir. Per altra banda, s'han implementat els coneixements assolits en Processament d'imatge biomèdiques per a poder entendre les comandes de skimage i matplotlib que s'han aplicat a les imatges, molt semblants a les de Matlab, però en llenguatge Python.

Gràcies a aquest projecte he après a interaccionar amb professionals del món de la recerca i la medicina per a establir els requisits inicials de l'usuari, a utilitzar la metodologies en cascada per al desenvolupament de software, i la programació orientada a objectes. Així també, el disseny previ a la implementació amb diferents diagrames UML per als casos d'ús, l'estructura de dades i el disseny gràfic. I per últim la realització de la documentació necessària per a la instal·lació, utilització i desenvolupament de software.

Bibliografía

Anònim. 2019. How to Create Pandas DataFrame in Python. *Data to Fish.* [En línea] 03 de 04 de 2019. [Citado el: 04 de 03 de 2019.] https://datatofish.com/create-pandas-dataframe/.

—. 2005. Learn PyQt programming language. *Tutorials Point.* [En línea] 2005. [Citado el: 30 de 02 de 2019.] https://www.tutorialspoint.com/pyqt/index.htm.

—. 2017. PyQt. *Python Programming Language.* [En línea] 07 de 2017. [Citado el: 2 de 03 de 2019.] https://pythonprogramminglanguage.com/pyqt/.

—. 2016. PyQt5 - Python Tutorial. *Pythonspot.* [En línea] 26 de 07 de 2016. [Citado el: 01 de 03 de 2019.] https://pythonspot.com/pyqt5/.

—. Python Classes and Objects. *W3schools.* [En línea] [Citado el: 30 de 02 de 2019.] https://www.w3schools.com/python/python_classes.asp.

COEIC/AEIC, Subcomissió de Recursos Humans de la Comissió de Gestió Empresarial del. 2017. Fulls dels Enginyers. [En línea] 29 de 11 de 2017. [Citado el: 21 de 05 de 2019.] http://fullsdelsenginyers.cat/article/primers-salaris-joves-enginyers.

Corporació Catalana de Mitjans Audiovisuals, SA. 2019. TV3 La Marató. [En línea] 2019. [Citado el: 20 de 05 de 2019.] https://www.ccma.cat/tv3/marato/2019/230/.

Electrocalculator. 2018. Electrocalculator. [En línea] 2018. [Citado el: 22 de 05 de 2019.] https://www.electrocalculator.com/avanzado.php.

FEDER. 2019. Federación Española de Enfermedades Raras. [En línea] 2019. [Citado el: 20 de 05 de 2019.] https://enfermedades-raras.org/index.php/enfermedades-raras.

García, Marcel. 2015. Xtec. *Desenvolupament d'interfícies.* [En línea] 2015. [Citado el: 21 de 05 de 2019.] https://ioc.xtec.cat/materials/FP/Materials/2252_DAM/DAM_2252_M07/web/html/media/fp_dam_m07_u1_pdfindex.pdf.

—. 2012. Xtec. *Introducció al disseny orientat a objectes.* [En línea] 2012. [Citado el: 21 de 05 de 2019.] http://ioc.xtec.cat/materials/FP/Materials/2252_DAM/DAM_2252_M05/web/html/media/fp_dam_m05_u3_pdfindex.pdf.

2012. Immage processing and analysis in Java. *imaeJ.* [En línea] 06 de 12 de 2012. https://imagej.nih.gov/ij/docs/intro.html.

John Hunter, Darren Dale, Eric Firing, Michael Droettboom i l'equip de desenvolupament de Matplotlib. 2012. Matplotlib Documentation. *Matplotlib.* [En línea] 2012. [Citado el: 05 de 03 de 2019.] https://matplotlib.org/.

López, José M. Drake i Patricia. 2017. CTR Universidad de cantabria. [En línea] 2017. [Citado el: 21 de 05 de 2019.] https://www.ctr.unican.es/asignaturas/Ingenieria_Software_4_F/Doc/M7_09_VerificacionValidacion-2011.pdf.

Selectra. 2018. Tarifasgasluz. [En línea] 2018. [Citado el: 22 de 05 de 2019.] https://tarifasgasluz.com/faq/precio-kwh.

Tejerina, Martín. 2011. Programación Orientada a Objetos . *monografias.com.* [En línea] Monografias.com S.A., 06 de 2011. https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion1/ventajas_desventajas.html.

VAN DER WALT, Stefan, et al. scikit-image: image processing in Python. PeerJ, 2014, 2: e453.

Varis. 2004. *Stack Overflow.* [En línea] 2004. [Citado el: 02 de 03 de 2019.] https://stackoverflow.com/.

—. 2015. Forum Qt. *Qt.* [En línea] 2015. [Citado el: 02 de 03 de 2019.] https://forum.qt.io/.

—. 2012. Image Processing in Pyhton. *Scikit-image.* [En línea] 2012. [Citado el: 02 de 03 de 2019.] https://scikit-image.org/.

Waskom, Michael. 2016. Statistical data visualization. *Seaborn.* [En línea] 2016. [Citado el: 05 de 03 de 2019.] https://seaborn.pydata.org/.

Yllanes, Edwin Christian. 2018. GitHub. [En línea] 9 de 11 de 2018. [Citado el: 26 de 05 de 2019.] https://github.com/eyllanesc/stackoverflow/blob/master/questions/44603119/PandasModel.py.

Annex

En l'annex s'adjunta el codi de cada script:

Script project.py

```
import sys, os
from PyQt5.QtWidgets import *

import scene
from plot_scatter import PlotScat
from centralwidget import CentralWidget



class MainWindow(QMainWindow):
    def __init__(self, *args):
        super().__init__(*args)
        self.plot_scatter=PlotScat()
        self.scene = scene.Scene()
        self.plot_scatter=PlotScat()
        self.initUI()
        self.filename=None

    def initUI(self):
        self.setWindowTitle("Processament i anàlisi d'imatges biomèdiques")
        self.setGeometry(0, 0, 1300, 700)

        self.create_menu()
        self.central = CentralWidget()
        self.setCentralWidget(self.central)
        self.show()
```

```
def create_menu(self):
    self.menubar = self.menuBar()

    ###
    self.menuFile = self.menubar.addMenu('Fitxer')

    readDataAction = QAction( 'Seleccionar imatge', self)
    readDataAction.setShortcut('Ctrl+F')
    readDataAction.triggered.connect(self.readImage)
    self.menuFile.addAction(readDataAction)

    exitAction = QAction('Sortir', self)
    exitAction.setShortcut('Ctrl+Q')
    exitAction.setStatusTip('Sortir de l\'aplicació')
    exitAction.triggered.connect(self.close)
    self.menuFile.addAction(exitAction)



    ###
    self.menuPros = self.menubar.addMenu('Processament')

    createProcess = QAction('Contrast gamma', self)
    createProcess.triggered.connect(self.gammacontrast)
    self.menuPros.addAction(createProcess)

    createProcess = QAction('Contrast log', self)
    createProcess.triggered.connect(self.logcontrast)
    self.menuPros.addAction(createProcess)

    self.menuCreate = self.menubar.addMenu('Segmentació')

    createAction = QAction('Otsu', self)
    createAction.triggered.connect(self.otsu_thresh)
    self.menuCreate.addAction(createAction)
```

```
secondAction = QAction('Watershed', self)
secondAction.triggered.connect(self.watershed)
self.menuCreate.addAction(secondAction)

thirdAction = QAction('K-means', self)
thirdAction.triggered.connect(self.kmean)
self.menuCreate.addAction(thirdAction)

fourthAction = QAction('Felzenszwalb', self)
fourthAction.triggered.connect(self.felzen)
self.menuCreate.addAction(fourthAction)


self.menuTable = self.menubar.addMenu('Taula')

tableAction = QAction('Propietats d\'Otsu', self)
tableAction.triggered.connect(self.DisplayProp)
self.menuTable.addAction(tableAction)

wtableAction = QAction('Propietats de watershed', self)
wtableAction.triggered.connect(self.Displaywat)
self.menuTable.addAction(wtableAction)

ktableAction = QAction('Propitats de k-means', self)
ktableAction.triggered.connect(self.Displayk)
self.menuTable.addAction(ktableAction)

ftableAction = QAction('Propietats de felzenszwalb', self)
ftableAction.triggered.connect(self.Displayf)
self.menuTable.addAction(ftableAction)


self.menuPlot = self.menubar.addMenu('Gràfic')
```

```
        plotAction = QAction('Area/Intensitat', self)
        plotAction.triggered.connect(self.scatterplot)
        self.menuPlot.addAction(plotAction)



        self.menuHelp = self.menubar.addMenu('Ajuda')
        aboutAction = QAction('Sobre el programa', self)
        aboutAction.triggered.connect(self.about)
        self.menuHelp.addAction(aboutAction)



        self.statusBar().showMessage('Preparat')
        self.show()

    def readImage(self) :
        options = QFileDialog.Options()
        self.filename, ok = QFileDialog.getOpenFileName(self,"Open file", "","Images (*.png *.tiff *.tif
*.jpg)", options=options)
        if ok :
            self.scene.read_image(self.filename)
            ima = self.scene.images
            nom=self.scene.nomimages
            self.central.render_image(ima)



    def about(self) :
        text = '''<center> Aquesta és la última versió del programa disenyat en el TFG,
        l\'objectiu del qual és visualitzar, processar i analítzar imatges amb diferents tipus de preocessat
        i segmentació.<br><center>Sandra Bernaus<br><br>UPC-EEBE<br><br>Course 2018-
2019<center>'''
        msgBox = QMessageBox.about(self, "Medical Images Project", text)


    def closeEvent(self, event):
```

```
        reply = QMessageBox.question(self, 'Missatge',
                                     "Està segur de voler sortir?", QMessageBox.Yes|
                                     QMessageBox.No, QMessageBox.No)



        if reply == QMessageBox.Yes:
            event.accept()
        else:
            event.ignore()



    def gammacontrast(self):
        if not self.filename is None and self.filename != '':#sel.filename='' when selecting file cancel button is pressed, then an error comes up
            self.scene.gammacorrection(self.filename)
            ima=self.scene.process
            self.central.render_seg(ima)


    def logcontrast(self):
        if not self.filename is None and self.filename != '':#sel.filename='' when selecting file cancel button is pressed, then an error comes up
            self.scene.logcorrection(self.filename)
            ima=self.scene.process
            self.central.render_seg(ima)


    def otsu_thresh(self) :
        if not self.filename is None and self.filename != '':
            self.scene.read_seg()
            dataseg = self.scene.segmen
            self.central.render_seg(dataseg)


    def watershed(self):
        if not self.filename is None and self.filename != '':
            self.scene.watersheds()
```

```
        dataseg = self.scene.segmen
        self.central.render_seg(dataseg)

    def kmean(self):
        if not self.filename is None and self.filename != '':
            self.scene.kmeans()
            dataseg = self.scene.segmen
            self.central.render_seg(dataseg)

    def felzen(self):
        if not self.filename is None and self.filename != '':
            self.scene.felzens()
            dataseg = self.scene.segmen
            self.central.render_seg(dataseg)

    def DisplayProp(self):
        if not self.filename is None and self.filename != '':
            self.scene.read_props()
            propietats = self.scene.data
            self.central.render_prop(propietats)

    def Displaywat(self):
        if not self.filename is None and self.filename != '':
            self.scene.read_propsw()
            propietats = self.scene.data
            self.central.render_prop(propietats)

    def Displayk(self):
        if not self.filename is None and self.filename != '':
            self.scene.read_propsk()
            propietats = self.scene.data
            self.central.render_prop(propietats)

    def Displayf(self):
```

```
        if not self.filename is None and self.filename != '':
            self.scene.read_propsf()
            propietats = self.scene.data
            self.central.render_prop(propietats)

    def scatterplot(self):
        if not self.filename is None and self.filename != '':
            self.scene.read_props()
            propietats = self.scene.data
            self.plot_scatter.render_scat(propietats)



if __name__=='__main__':
    app = QApplication(sys.argv)
    p = MainWindow()
    sys.exit(app.exec_())
            self.central.render_prop(propietats)

    def Displaywat(self):
        if not self.filename is None and self.filename != '':
            self.scene.read_propsw()
            propietats = self.scene.data
            self.central.render_prop(propietats)

    def Displayk(self):
        if not self.filename is None and self.filename != '':
            self.scene.read_propsk()
            propietats = self.scene.data
            self.central.render_prop(propietats)

    def Displayf(self):
        if not self.filename is None and self.filename != '':
            self.scene.read_propsf()
            propietats = self.scene.data
```

```
        self.central.render_prop(propietats)


    def scatterplot(self):
        if not self.filename is None and self.filename != '':
            self.scene.read_props()
            propietats = self.scene.data
            self.plot_scatter.render_scat(propietats)



if __name__=='__main__':
    app = QApplication(sys.argv)
    p = MainWindow()
    sys.exit(app.exec_())
```

Script centerwidget.py

```
from PyQt5.QtWidgets import *
from matplotlib.backends.backend_qt4agg import NavigationToolbar2QT as NavigationToolbar
import scene

from plot_scatter import PlotScat
from plot_canvas import PlotCanvas
from table_widget import TableWidget



class CentralWidget(QWidget) :
    def __init__(self, *args):
        super().__init__(*args)
        self.initUI()

    def initUI(self):

        self.plot_canvas = PlotCanvas()
```

```
        self.toolbar = NavigationToolbar(self.plot_canvas,self)
        self.table_widget = TableWidget()

        #self.toolbars = NavigationToolbar(self.plot_scatter,self)



        self.vbox = QVBoxLayout()
        self.vbox.addWidget(self.toolbar)
        self.vbox.addWidget(self.plot_canvas)



        self.hbox = QHBoxLayout()
        self.hbox.addLayout(self.vbox)
        self.hbox.addWidget(self.table_widget)
        self.setLayout(self.hbox)

    def render_image(self, ima) :
        self.plot_canvas.render_image(ima)
        self.parent().statusBar().showMessage('Imatge mostrada')

    def render_seg(self, dataseg):
        self.plot_canvas.render_seg(dataseg)
        self.parent().statusBar().showMessage('Imatge processada')

    def render_prop(self, propietats):
        self.table_widget.render_prop(propietats)
        self.parent().statusBar().showMessage('Propietats mostrades en la taula')
```

Script plot_canvas.py

```
from PyQt5.QtWidgets import QSizePolicy
from PyQt5.QtWidgets import *
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.backends.backend_qt4agg import NavigationToolbar2QT as NavigationToolbar
from matplotlib.figure import Figure
import matplotlib.pyplot as plt
from mpldatacursor import datacursor


class PlotCanvas(FigureCanvas):

  def __init__(self, parent=None, width=12, height=12, dpi=130):
    fig = Figure(figsize=(width, height), dpi=dpi)
    super().__init__(fig)
    self.setParent(parent)
    FigureCanvas.setSizePolicy(self,
        QSizePolicy.Expanding,
        QSizePolicy.Expanding)
    FigureCanvas.updateGeometry(self)


    fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(8, 3))
    ax = axes.ravel()
    fig.set_tight_layout(True)

    self.ax1 = self.figure.add_subplot(121)
    self.ax2 = self.figure.add_subplot(122, sharex=self.ax1, sharey=self.ax1)

  def render_image(self, ima):
    self.ax1.clear()
```

```
        self.ax2.clear()
        self.ax1.imshow(ima)
        dc2 = datacursor(self.ax1, bbox=dict(alpha=1, fc='w'))#hover per mostrar cursor quan passi per
sobre
        self.ax1.set_title('Imatge')
        self.draw()



    def render_seg(self, dataseg):
        self.ax2.clear()
        self.ax2.imshow(dataseg)
        dc2 = datacursor(self.ax2, bbox=dict(alpha=1, fc='w'))
        self.ax2.set_title('Processada')
        self.draw()
```

```
Script table_widget.py
import sys, csv, os
from PyQt5.QtWidgets import *
from PyQt5 import QtCore
from pandas import DataFrame

from pandasmodel import PandasModel

class TableWidget(QWidget):
   def __init__(self):
      QWidget.__init__(self)
      self.setFixedSize(400, 500)
      self.table = QTableView(self)
      self.buttonSave = QPushButton('Guardar', self)
      self.buttonSave.clicked.connect(self.handleSave)

      layout = QVBoxLayout(self)
      layout.addWidget(self.table)
      layout.addWidget(self.buttonSave)

   def render_prop(self, propietats):
      self.model=PandasModel(propietats)
      self.table.setModel(self.model)
      self.data=propietats



   def handleSave(self):
      path, ok = QFileDialog.getSaveFileName(
            self, 'Guardar Fitxer', '', 'CSV(*.csv)')

      if ok:
         self.data.to_csv(path, index = None, header=True)
```

*Script pandasmodel.py (dissenyat per **Yllanes, Edwin Christian**)*

```
from PyQt5 import QtCore

import pandas as pd


class PandasModel(QtCore.QAbstractTableModel):
    def __init__(self, df = pd.DataFrame(), parent=None):
        QtCore.QAbstractTableModel.__init__(self, parent=parent)
        self._df = df

    def headerData(self, section, orientation, role=QtCore.Qt.DisplayRole):
        if role != QtCore.Qt.DisplayRole:
            return QtCore.QVariant()

        if orientation == QtCore.Qt.Horizontal:
            try:
                return self._df.columns.tolist()[section]
            except (IndexError, ):
                return QtCore.QVariant()
        elif orientation == QtCore.Qt.Vertical:
            try:
                # return self.df.index.tolist()
                return self._df.index.tolist()[section]
            except (IndexError, ):
                return QtCore.QVariant()

    def data(self, index, role=QtCore.Qt.DisplayRole):
        if role != QtCore.Qt.DisplayRole:
            return QtCore.QVariant()

        if not index.isValid():
```

```
            return QtCore.QVariant()

        return QtCore.QVariant(str(self._df.ix[index.row(), index.column()]))

    def setData(self, index, value, role):
        row = self._df.index[index.row()]
        col = self._df.columns[index.column()]
        if hasattr(value, 'toPyObject'):
            # PyQt4 gets a QVariant
            value = value.toPyObject()
        else:
            # PySide gets an unicode
            dtype = self._df[col].dtype
            if dtype != object:
                value = None if value == '' else dtype.type(value)
        self._df.set_value(row, col, value)
        return True

    def rowCount(self, parent=QtCore.QModelIndex()):
        return len(self._df.index)

    def columnCount(self, parent=QtCore.QModelIndex()):
        return len(self._df.columns)

    def sort(self, column, order):
        colname = self._df.columns.tolist()[column]
        self.layoutAboutToBeChanged.emit()
        self._df.sort_values(colname, ascending= order == QtCore.Qt.AscendingOrder, inplace=True)
        self._df.reset_index(inplace=True, drop=True)
        self.layoutChanged.emit()
```

Script plot_scatter.py

```
from PyQt5.QtWidgets import *
import matplotlib.pyplot as plt
import seaborn as sns

class PlotScat(QWidget):
    def __init__(self):
        QWidget.__init__(self)



    def render_scat(self, propietats):
        fig = plt.figure(1)
        fig.canvas.set_window_title('Gràfic Scatter')
        plt.subplot(111)
        sns.set(style="ticks")
        sns.scatterplot(x=propietats['Intensitat'],y=propietats['Area'])
        plt.show()
```

Script scene.py

```
from segmentada import Segmentada
from regionprops import RegionProps

class Scene():

   def __init__(self):
      self.images=None
      self.nomimages=None
      self.process=None
      self.segmen=None
      self.data=None

   def read_image(self, filename):
      ima = Segmentada()
      ima.from_image(filename)
      self.images=ima.imaori
      self.nomimages=ima.nom

   def gammacorrection(self,filename):
      ima= Segmentada()
      ima.gammacorrection(filename)
      self.images=ima.imaori
      self.process=ima.process

   def logcorrection(self,filename):
      ima= Segmentada()
      ima.logcorrection(filename)
      self.images=ima.imaori
      self.process=ima.process
```

```
def read_seg(self):
    dataseg = Segmentada()
    dataseg.segmentacio(self.images)
    self.segmen=dataseg.segmen

def watersheds(self):
    dataseg = Segmentada()
    dataseg.segmentacio_water(self.images)
    self.segmen=dataseg.segmen

def kmeans(self):
    dataseg = Segmentada()
    dataseg.segmentacio_k(self.images)
    self.segmen=dataseg.segmen

def felzens(self):
    dataseg = Segmentada()
    dataseg.segmentacio_felz(self.images)
    self.segmen=dataseg.segmen

def read_props(self):
    propietats=RegionProps()
    propietats.regseg(self.images)
    self.data=propietats.data

def read_propsw(self):
    propietats=RegionProps()
    propietats.regwat(self.images)
    self.data=propietats.data

def read_propsk(self):
    propietats=RegionProps()
    propietats.regkmeans(self.images)
    self.data=propietats.data
```

```
  def read_propsf(self):
      propietats=RegionProps()
      propietats.regfelz(self.images)
      self.data=propietats.data
```

Script segmentada.py

```
from skimage import io, exposure
from skimage.filters import threshold_otsu, sobel
from skimage.segmentation import clear_border, watershed, felzenszwalb, slic
from skimage.measure import label
from skimage.morphology import closing, square
from skimage.transform import resize
from skimage.color import label2rgb, rgb2gray
class Segmentada:

   def __init__(self, imaori=None,ima=None, nomim=None,proces=None,label_image=None,
dataseg=None):
      self.imaori = imaori
      self.nom=nomim
      self.segmen = dataseg
      self.process= proces
      self.label=label_image

   def from_image(self, filename):
      self.imaori = io.imread(filename)
      self.nom = filename[:filename.index('.')]

   def gammacorrection(self, filename):
      self.imaori=rgb2gray(io.imread(filename))
      self.process = exposure.adjust_gamma(self.imaori, 2)
      self.reima()
```

```
def logcorrection(self, filename):
    self.imaori=rgb2gray(io.imread(filename))
    self.process = exposure.adjust_log(self.imaori, 1)
    self.reima()

def reima(self):
    if self.process is not None:
        self.imaori=self.process

def segmentacio(self, image):
    self.imaori=rgb2gray(image)#añadir filename en variables
    self.thresh = threshold_otsu(self.imaori)
    bw = closing(self.imaori < self.thresh, square(3))
    cleared = clear_border(bw)
    self.label = label(cleared)
    self.segmen = label2rgb(self.label, image=self.imaori)

def segmentacio_water(self,image): #añadir filename en variables
    self.imaori=rgb2gray(image)#añadir filename en variables
    gradient = sobel(rgb2gray(self.imaori))
    self.label = watershed(gradient, markers=600)
    self.segmen=label2rgb(self.label, self.imaori, kind='avg')

def segmentacio_k(self,image): #añadir filename en variables
    self.imaori=rgb2gray(image)#añadir filename en variables
    self.label = slic(self.imaori, compactness=0.2, n_segments=600)
    self.segmen = label2rgb(self.label, self.imaori, kind='avg')

def segmentacio_felz(self,image):
    self.imaori=rgb2gray(image)#añadir filename en variables
    self.label = felzenszwalb(self.imaori, scale=100, sigma=0.5, min_size=50)
    self.segmen = label2rgb(self.label, self.imaori, kind='avg')
```

Script regionprops.py

```
import pandas as pd

from skimage.measure import regionprops

from segmentada import Segmentada

class RegionProps:
        def __init__(self, dataframe=None, labelarea=None, labelcentro=None, labellabel=None,
scat=None ):
                self.data=dataframe
                self.areas=labelarea
                self.centroids=labelcentro
                self.labels=labellabel



        def regseg(self,image):
                dataseg=Segmentada()
                dataseg.segmentacio(image)
                self.regions=regionprops(dataseg.label,intensity_image=dataseg.imaori)
                self.areas = [prop.area for prop in self.regions]
                self.centroids=[prop.centroid for prop in self.regions]
                self.labels= [prop.label for prop in self.regions]
                self.inte= [prop.mean_intensity for prop in self.regions]
                self.list={'Etiqueta': self.labels,'Area':self.areas,'Centroide': self.centroids,
'Intensitat':self.inte}
                self.data=pd.DataFrame(self.list, columns=['Etiqueta','Area','Centroide','Intensitat'])


        def regwat(self,image):
                dataseg=Segmentada()
                dataseg.segmentacio_water(image)
                self.regions=regionprops(dataseg.label, intensity_image=dataseg.imaori)
```

```
                self.areas = [prop.area for prop in self.regions]
                self.centroids=[prop.centroid for prop in self.regions]
                self.labels= [prop.label for prop in self.regions]
                self.inte= [prop.mean_intensity for prop in self.regions]
                self.list={'Etiqueta': self.labels,'Area':self.areas,'Centroide': self.centroids,
'Intensitat':self.inte}
                self.data=pd.DataFrame(self.list, columns=['Etiqueta','Area','Centroide','Intensitat'])


        def regkmeans(self,image):
                dataseg=Segmentada()
                dataseg.segmentacio_k(image)
                self.regions=regionprops(dataseg.label,intensity_image=dataseg.imaori)
                self.areas = [prop.area for prop in self.regions]
                self.centroids=[prop.centroid for prop in self.regions]
                self.labels= [prop.label for prop in self.regions]
                self.inte= [prop.mean_intensity for prop in self.regions]
                self.list={'Etiqueta': self.labels,'Area':self.areas,'Centroide': self.centroids,
'Intensitat':self.inte}
                self.data=pd.DataFrame(self.list, columns=['Etiqueta','Area','Centroide','Intensitat'])



        def regfelz(self,image):
                dataseg=Segmentada()
                dataseg.segmentacio_felz(image)
                self.regions=regionprops(dataseg.label,intensity_image=dataseg.imaori)
                self.areas = [prop.area for prop in self.regions]
                self.centroids=[prop.centroid for prop in self.regions]
                self.labels= [prop.label for prop in self.regions]
                self.inte= [prop.mean_intensity for prop in self.regions]
                self.list={'Etiqueta': self.labels,'Area':self.areas,'Centroide': self.centroids,
'Intensitat':self.inte}
                self.data=pd.DataFrame(self.list, columns=['Etiqueta','Area','Centroide','Intensitat'])
```